Universidade Estadual de Campinas
Instituto de Computação

Daniel Guimarães do Lago

Algoritmos de Escalonamento de Máquinas Virtuais Cientes de Topologia e Energia em Data Centers

CAMPINAS
2018

Daniel Guimarães do Lago

Algoritmos de Escalonamento de Máquinas Virtuais Cientes de Topologia e Energia em Data Centers

Tese apresentada ao Instituto de Computação da Universidade Estadual de Campinas como parte dos requisitos para a obtenção do título de Doutor em Ciência da Computação.

Orientador: Prof. Dr. Edmundo Roberto Mauro Madeira

Este exemplar corresponde à versão final da Tese defendida por Daniel Guimarães do Lago e orientada pelo Prof. Dr. Edmundo Roberto Mauro Madeira.

CAMPINAS
2018

Agência(s) de fomento e nº(s) de processo(s): CAPES; CNPq, 167384/2014-7
ORCID: https://orcid.org/0000-0002-7793-2266

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca do Instituto de Matemática, Estatística e Computação Científica
Ana Regina Machado - CRB 8/5467

Lago, Daniel Guimarães do, 1985-
L137a Algoritmos de escalonamento de máquinas virtuais cientes de topologia e energia em data centers / Daniel Guimarães do Lago. – Campinas, SP : [s.n.], 2018.

Orientador: Edmundo Roberto Mauro Madeira.
Tese (doutorado) – Universidade Estadual de Campinas, Instituto de Computação.

1. Computação em nuvem. 2. Escalonamento. 3. Máquinas virtuais. 4. Eficiência energética. 5. Topologia. I. Madeira, Edmundo Roberto Mauro, 1958-. II. Universidade Estadual de Campinas. Instituto de Computação. III. Título.

Informações para Biblioteca Digital

Título em outro idioma: Topology and energy aware virtual machine scheduling algorithms in data centers
Palavras-chave em inglês:
Cloud computing
Scheduling
Virtual machines
Energy efficiency
Topology
Área de concentração: Ciência da Computação
Titulação: Doutor em Ciência da Computação
Banca examinadora:
Edmundo Roberto Mauro Madeira [Orientador]
Djamel Fawzi Hadj Sadok
Fabio Luciano Verdi
Christian Esteve Rothenberg
Nelson Luis Saldanha da Fonseca
Data de defesa: 28-02-2018
Programa de Pós-Graduação: Ciência da Computação

Universidade Estadual de Campinas
Instituto de Computação

Daniel Guimarães do Lago

Algoritmos de Escalonamento de Máquinas Virtuais Cientes de Topologia e Energia em Data Centers

Banca Examinadora:

- Prof. Dr. Edmundo Roberto Mauro Madeira
  UNICAMP-IC
- Prof. Dr. Djamel Fawzi Hadj Sadok
  UFPE
- Prof. Dr. Fabio Luciano Verdi
  UFSCar
- Prof. Dr. Christian Esteve Rothenberg
  UNICAMP-FEEC
- Prof. Dr. Nelson Luis Saldanha da Fonseca
  UNICAMP-IC

A ata da defesa com as respectivas assinaturas dos membros da banca encontra-se no processo de vida acadêmica do aluno.

Campinas, 28 de fevereiro de 2018

Agradecimentos

Agradeço à minha família pelo apoio. Em especial à mamãe Vânia, ao tio Mané, à vovó Teresinha, à vovó Lolita (*in memorian*), ao meu avô Tomé (*in memorian*), pelo incentivo. Aos meus irmãos Davi, Lucas, Luísa e André, pelo companheirismo. À minha noiva Mayara, pelo amor, carinho e compreensão. À dona Iandra e ao senhor Lajara, pelo suporte.

À Unicamp e aos professores do IC por me possibilitarem um crescimento tão importante. Aos meus colegas do LRC, pela compreensão. Aos meus amigos de república, Júlio e Danilo, pela amizade.

Aos professores do CEFET-MG, em especial os da Coordenação de Informática, pelo apoio.

Aos técnicos administrativos do IC, pela atenção e eficiência.

Ao professor Deep Medhi, pela colaboração de fundamental importância no desenvolvimento deste trabalho.

Em especial ao meu orientador, professor Edmundo, que me mostrou os longínquos limites da excelência.

À CAPES, ao CNPq, à Unicamp, à FAPESP e ao CEFET-MG pelo apoio financeiro que possibilitou o desenvolvimento deste trabalho.

Resumo

O crescente consumo de energia e a poluição gerada para alimentarem a infraestrutura da computação em nuvem, impulsionados pela proliferação de dispositivos com baixo poder de processamento e a crescente necessidade de computação elástica, têm sido uma constante preocupação, refletida em numerosas abordagens no contexto da computação verde para lidar com este problema. Além disso, percebemos que o principal padrão escolhido para a conectividade dos *data centers* que operam em nuvem, *Ethernet*, tem apresentado rápido aumento das taxas de transmissão, bem como a expectativa de continuidade deste. Neste trabalho efetuamos estudos acerca dos impactos do aumento das taxas de transmissão no processo de escalonamento de máquinas virtuais, focando, em especial, no consumo de energia, no *makespan*, nos tempos de execução de cargas de trabalho e no número de migrações de máquinas virtuais em *data centers* que operam em nuvens, no contexto da ciência de energia. Desenvolvemos, também, um modelo empírico para estimar o consumo de energia em função da largura de banda para um conjunto de cenários. Expandimos nossos estudos e apresentamos o `BALA`, um algoritmo de escalonamento de máquinas virtuais ciente de energia e de largura de banda, com especial aproveitamento em *data centers* com agrupamentos de servidores que apresentam heterogeneidade em largura de banda. Finalmente, apresentamos o `TEA`, um algoritmo de escalonamento de máquinas virtuais ciente de energia e de topologia de rede, que considera não somente os elementos de borda, mas também o núcleo da rede, sendo um algoritmo escalável capaz de atuar em uma diversidade de cenários, incluindo, mas não se limitando, a nuvens constituídas de *data centers* geodistribuídos ou não, com topologias de rede arbitrárias, que admitem ou não a migração de máquinas virtuais, homogêneos ou heterogêneos tanto em servidores quanto no núcleo da rede, podendo suportar SLA com garantia de processamento para máquinas virtuais e dar preferência às máquinas virtuais de prioridades elevadas. Para atingir nossos objetivos, utilizamos em ambos algoritmos técnicas como o desligamento de servidores ociosos, DVFS e a migração de máquinas virtuais. Em adição, no `TEA`, também empregamos um conceito de maximização de eficiência em energia de rotas e o desligamento de comutadores de pacotes ociosos. Resultados obtidos por simulação em um extenso número de cenários mostram que este algoritmo, confrontado a diversos outros algoritmos cientes de energia, apresenta os melhores resultados em economia de energia em aproximadamente metade dos casos, e o melhor *makespan* na maior parte dos casos. Os ganhos observados são notáveis em *data centers* geodistribuídos, heterogêneos e com topologias constituídas por um número grande de comutadores de pacotes.

Abstract

The increasing energy consumption and the pollution generated to power the infrastructure of the cloud computing, driven by the proliferation of devices with low processing power and the increasing need for elastic computing, have been a constant concern, reflected in numerous approaches in the context of green computing to deal with this problem. In addition, we realize that the connectivity leading choice for data centers that operates in the cloud, Ethernet, has shown rapid increase in transmission rates, as well as the expectation of continuity of this growth. In this work, we study the impacts of the increase of transmission rates in the virtual machine scheduling process, focusing in particular on power consumption, makespan, workload execution times, and the number of machine migrations in data centers that operate in clouds in the context of energy awareness. We have also developed an empirical model to estimate energy consumption as a function of bandwidth for a set of scenarios. We have expanded our studies and presented the `BALA`, an energy-aware and bandwidth-aware scheduling algorithm, with special use in data centers with server groups having heterogeneity in bandwidth. Finally, we present the `TEA`, a virtual scheduling algorithm aware of energy and network topology, which considers not only the edge elements, but also the network core, being a scalable algorithm capable of acting in a diversity of scenarios, including, but not limited to, clouds made up of geo-distributed or non geo-distributed data centers with arbitrary network topologies that allow or disallow the migration of virtual machines, homogeneous or heterogeneous servers and network core devices, and supporting SLA with guaranteed processing for virtual machines and giving preference for high priority virtual machines. To achieve our goals, we use in both algorithms techniques such as shutdown of idle servers, DVFS, and migration of virtual machines. In addition, in `TEA`, we also employ a concept of maximizing energy efficiency in routes and shutting down idle switches. Results obtained by simulation in an extensive number of scenarios show that this algorithm, compared to several other energy-aware algorithms, presents the best results in energy savings in approximately half of the cases, and the best makespan in most cases. The observed gains are notable in geo-distributed data centers, with topologies consisting of a large number of switches, and in heterogeneity cases.

Lista de Figuras

Lista de Tabelas

Sumário

Capítulo 1

Introdução

A computação em nuvem refere-se a um modelo de computação iniciado em 1999 por Fredrik Malmer (WebOS) [15] e que teve um rápido desenvolvimento a partir de 2010 [83], se tornando uma importante área de pesquisa nos últimos anos. De fato, é possível observar no Google Scholar centenas de milhares de trabalhos publicados sobre este modelo — com um aumento de mais de 1.400% no número de artigos publicados desde 2009 [57] — que permite acesso em rede ubíquo, conveniente e sob demanda para uma gama de recursos computacionais configuráveis (ex: redes, servidores, armazenamento, aplicações e serviços), que podem ser rapidamente aprovisionados e liberados com mínimo esforço administrativo ou interação do provedor de serviços [84].

Este modelo possui cinco características essenciais — auto-serviço sob demanda, acesso banda larga à rede, aprovisionamento dinâmico de recursos, elasticidade rápida e serviços — e em nível de infraestrutura, plataforma, desenvolvimento e software [6] — pago pelo uso. Alguns trabalhos vão além e propõem "tudo" como um serviço (XaaS) [92]. Para tornar tais serviços possíveis, recursos virtualizados, em especial máquinas virtuais (VMs), são frequentemente empregados em *data centers* computacionais [66] — conjuntos físicos de servidores — máquinas físicas — interconectados por uma rede, disponíveis para receberem máquinas virtuais e, consequentemente, cargas de trabalho. Uma máquina virtual é uma instância virtual de uma máquina, acessível através de uma nuvem, hospedada em um servidor determinado por um *broker* — um servidor responsável por lidar com eventos como requisição de recursos, criação de máquina virtual, retorno de cargas de trabalho e finalização de cargas de trabalho e máquinas virtuais. Um exemplo de tecnologia de *broker* é o *nova-scheduler* do OpenStack [95]. Uma máquina física pode hospedar múltiplas máquinas virtuais a depender da quantidade de recursos que essa dispõe e da quantidade de recursos esperados para as máquinas virtuais a serem alocadas.

Impulsionadas pelo surgimento de sucessivos dispositivos de baixo poder de processamento, várias plataformas para computação em nuvem surgiram/expandiram, notavelmente em grandes corporações — *Microsoft Azure* [86], *Google App Engine* [44], *Amazon EC2* [3] — corroborando a identificação da nuvem como uma mega-tendência [47].

Mas esta grande expansão também trouxe algumas preocupações. Koomey [68] afirmou que, nos EUA, o consumo de energia dos servidores é da mesma magnitude do consumo de energia de todas as televisões. De fato, Lefèvre [78] estimou que em 2009 todos os computadores do mundo usaram 10% da energia global consumida. Algumas

organizações em prol do meio ambiente questionam a computação em nuvem. O Greenpeace [45] estima que em 2020 o número de computadores pessoais ultrapassará 4 bilhões de dispositivos, e o número de dispositivos móveis duplicará, e para acompanhar o crescimento do número destes dispositivos também devem ocorrer o crescimento de *data centers* que operam em nuvens e, consequentemente, a energia necessária e poluição gerada para alimentação destes. Tang et al. afirmam que o custo de energia para manter *data centers* que operam em uma nuvem é vertiginosamente alto [115]. O Greenpeace estimou que em 2020 o consumo de energia requerida para a computação em nuvem atinja a casa dos 1,96 trilhões de kWh, além de uma preocupante emissão de 1.430 toneladas de dióxido de carbono por ano, conforme pode ser observado na Figura 1.1.

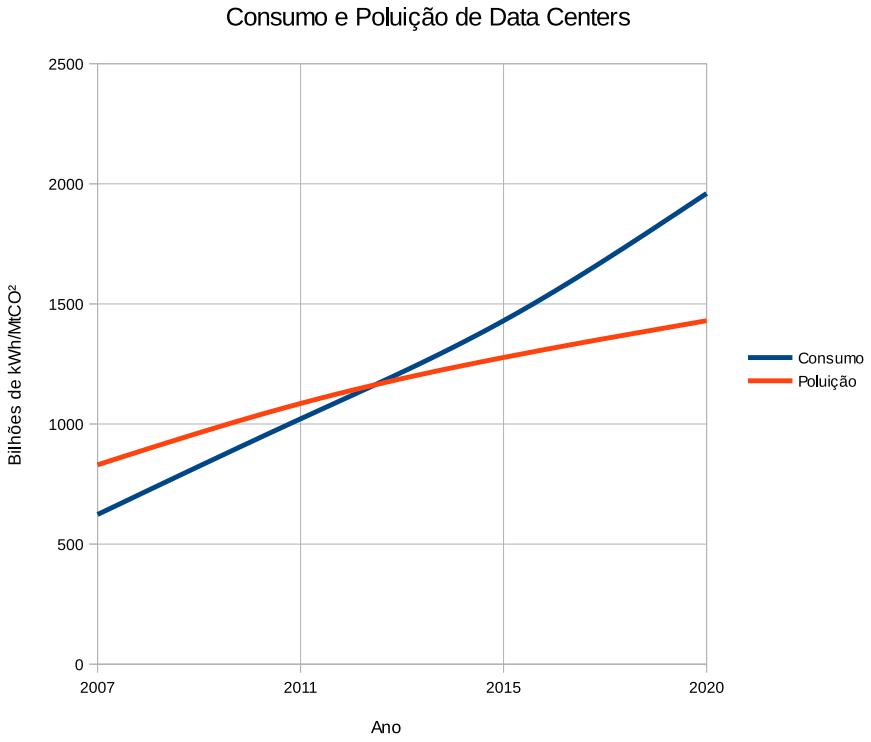

Figura 1.1: Estimativa de Consumo (Bilhões de kWh) e Emissões de MtCO2e na Computação em Nuvem

As preocupações ambientais têm surtido resultados positivos na evolução da eficiência em energia: se os níveis de eficiência em energia praticados em 2010 tivessem sido mantidos em 2014, os *data centers* dos EUA consumiriam aproximadamente 40 bilhões de kWh a mais. Mas ainda existe um longo caminho pela frente: estima-se que o crescimento do consumo de energia dos próximos 5 anos sejam os mesmos dos últimos 5 anos [113]. Com isto em mente, o papel da criação de infraestrutura para a computação em nuvem se torna uma área cada vez mais importante na pesquisa [60], em especial com o número de servidores empregados em *data centers* que operam em nuvem atingindo a marca de milhões [33].

A utilização de aplicativos dinâmicos cientes de energia pode tratar a subutilização de servidores bem como os crescentes custos de energia em um *data center* [29]. Entre as

diversas abordagens empregadas para se reduzir o consumo de energia nas nuvens computacionais está a adoção de algoritmos de escalonamento de máquinas virtuais cientes de energia [80]. Estes algoritmos são executados pelos *brokers* de *data centers* e objetivam determinar qual é o servidor que deverá hospedar uma dada máquina virtual, de modo a minimizar o consumo de energia do *data center*. Considerando que máquinas virtuais podem ser migradas entre máquinas físicas em uma rede intra-nuvem [24], tais algoritmos — mais do que atuar somente na seleção de uma máquina física para hospedar uma máquina virtual cuja instanciação está sendo solicitada — também possibilitam escolher uma melhor máquina física para uma máquina virtual que já está em execução, de modo que *brokers* determinem a migração de máquinas virtuais de máquinas físicas menos eficientes em energia para máquinas físicas mais eficientes, com possível posterior desligamento dessas. Pode-se ressaltar que migrações normalmente impactam em uma pequena degradação no desempenho de processamento das máquinas virtuais, no entanto, muitas vezes imperceptível ao usuário.

Um ponto importante para se considerar em *data centers* que operam na computação em nuvem é a interconexão da rede. A tecnologia mais comum para esta finalidade é a *Ethernet* [112, 114] que, nas últimas três décadas, apresentou um aumento das taxas de transmissão médio superior a 50% ao ano, evoluindo de 1 Mbps (802.3e) em 1987 [52] para 100 Gbps em 2015 (802.3bm) [54], conforme ilustrado na Figura 1.2. Além disso, o IEEE anunciou que existe demanda de taxa de transmissão de 10 Tbps para a *Ethernet* em 2020 [53].

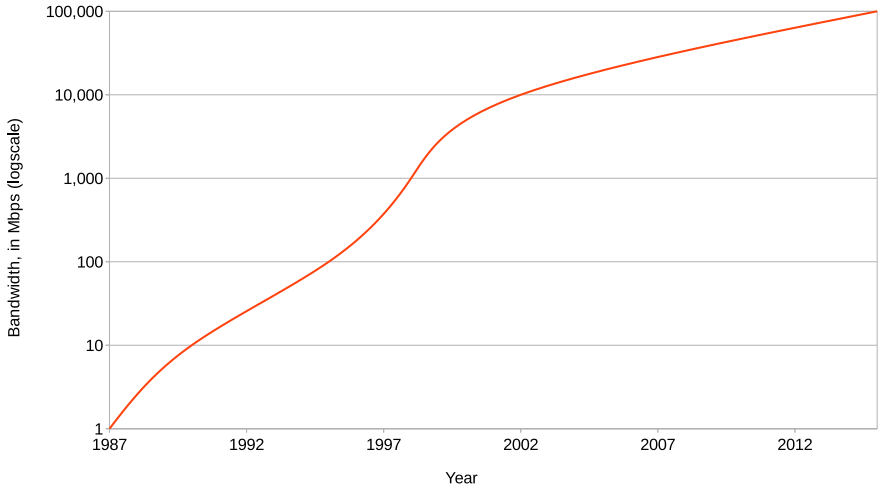

Figura 1.2: Evolução das Taxas de Transmissão da Tecnologia *Ethernet*

Nosso principal objetivo neste trabalho é propor um algoritmo de escalonamento de máquinas virtuais, ciente de energia e de topologia de rede, para minimizar o consumo de energia em *data centers* que operam em nuvem, contribuindo para os princípios da computação verde. Constatamos em [73] que redes mais rápidas podem reduzir o consumo de energia em nuvens cientes de energia. Taxas de transmissões maiores possibilitam a migração de máquinas virtuais em menor tempo, possibilitando desligamento mais rápido de servidores ociosos. Além disso, estas migrações mais rápidas também possibilitam a redução do *makespan* — tempo total compreendido desde o momento em que as cargas

de trabalho são recebidas pelo *data center* até o momento em que nos servidores são concluídas — e dos tempos de execução de cargas. Observamos também, em [74], que a perda de desempenho durante as migrações afeta menos o *makespan* do que o algoritmo de escalonamento de máquinas virtuais utilizado.

A implantação e a manutenção de *data centers* grandes apresentam elevados custos associados. Em virtude disso, frequentemente não são totalmente atualizados quando novas tecnologias computacionais — incluindo de redes — são lançadas. Do ponto de vista econômico, muitas vezes faz mais sentido atualizar partes do *data center*, o que acaba por criar *data centers* com máquinas de configurações heterogêneas, impactando na sua capacidade de processamento, RAM, largura de banda, etc. Consequentemente, isso pode gerar uma heterogeneidade, além das máquinas físicas em si, até mesmo da rede. Podemos observar essa heterogeneidade facilmente em nuvens que divulgam informações sobre suas máquinas físicas. No *Amazon EC2*, por exemplo, é possível observar uma heterogeneidade com pelo menos 5 grupos de máquinas físicas [4]: um grupo constituído por máquinas com processadores Xeon Platinum 8175, outro com Xeon E5-2686 v4, outro com Xeon E5-2676 v3, outro com Xeon E5-2670 v2 e outro com Xeon E5-2670. Neste exemplo, podemos observar a heterogeneidade: processadores diferentes, *sockets* (de processadores) diferentes, placas-mãe diferentes, etc., enfim, agrupamentos de servidores com configurações de hardware muito distintas. Além disso, a heterogeneidade das redes em *data centers* é um tópico que levanta interesse científico [108].

Em adição aos estudos apresentados, propomos neste trabalho algoritmos para o escalonamento ciente de energia de máquinas virtuais. Em um primeiro instante apresentamos o *Bandwidth-Aware Lago Allocator* (`BALA`), um algoritmo ciente de energia e de largura de banda. Este algoritmo é uma evolução de um predecessor, o *Lago Allocator* (`LA`), com adição do recurso de ciência de largura de banda com o intuito de melhorar seu desempenho no contexto de nuvens constituídas por *data centers* com heterogeneidade de largura de bandas em servidores.

Posteriormente, ampliamos o escopo, olhando não só para os elementos de borda, mas também para o núcleo da rede, considerando os estudos efetuados e os resultados obtidos com o `BALA`. Desenvolvemos com as técnicas observadas um novo algoritmo de escalonamento de máquinas virtuais — `TEA` — ciente de energia e da topologia de rede que integra os *data centers* de uma nuvem. Este algoritmo prioriza selecionar máquinas físicas mais eficientes em energia, mas também considerar aspectos da rede para determinar qual servidor deverá receber uma máquina virtual solicitada. Mais especificamente, o algoritmo que propomos tem como princípios fundamentais:

- Flexibilidade de Nuvens → atuação sem restrição de tipos de nuvens. Pode operar em nuvens privadas, públicas ou híbridas;
- Flexibilidade de *Data Centers* → atuação em nuvens constituídas tanto por um *data center* único quanto por *data centers* geodistribuídos; onde considera-se a constituição de uma geodistribuição de nuvem(ns) como *data centers* distribuídos geograficamente [103] e que compartilham recursos usando uma interface transparente aos usuários [30]. Tal geodistribuição pode ser implementada em diversos softwares para nuvens, por exemplo, usando o recurso de federação o *OpenNebula* [94].

Apresentamos na Figura 1.3 o modelo de geodistribuição de nuvem utilizado neste trabalho. Neste modelo, $dc0$ e $dc1$ representam *data centers* geograficamente distintos, interconectados pela internet, acessíveis pelos seus respectivos roteadores de borda;

- Flexibilidade de Topologia de Rede → atuação com ciência da topologia de rede, mas não dependente dela. Em outras palavras, o mesmo algoritmo é capaz de atuar em diversas topologias, como *Single-Hop Star*, *Flat-Tree*, *Binary Tree*, *K-Ary Fat Tree*, topologias arbitrárias, etc; além de considerar larguras de banda arbitrárias para quaisquer elementos de rede. Esta flexibilidade evita que o algoritmo se transforme em legado, podendo ser facilmente adaptado para possíveis arquiteturas futuras para *data centers*, como a vistas em [104];

- Maximização da Eficiência Energética da Rota → considerando a topologia, o algoritmo visa selecionar rotas com maior eficiência em energia considerando o trajeto fim-a-fim que as máquinas virtuais deverão percorrer — tanto para serem instanciadas, quanto para serem migradas. Esta eficiência em energia é diretamente proporcional à largura de banda da rota e inversamente proporcional à soma das potências dos comutadores de pacotes que compõem a rota;

- Determinação do Número de Conexões Simultâneas para Submissão e para Migração de Máquinas Virtuais → sendo executado por um *broker*, o algoritmo tem conhecimento do número de conexões que estão em andamento, seja para submeter uma máquina virtual a ser instanciada, seja para migrar máquina virtual entre servidores. O algoritmo permite que o administrador determine o número máximo de conexões para cada um destes tipos de conexões. Números muito baixos tendem a aumentar o *makespan*, enquanto números muito altos tendem a aumentar o consumo de energia, por demandar muitos servidores ligados durante o tempo da transferência das máquinas virtuais. Bons números vão depender dos sistemas finais onde as máquinas virtuais estão armazenadas e da largura de banda entre estes sistemas finais e os servidores que as hospedarão;

- Pré-processamento de máquinas virtuais a serem submetidas → o algoritmo executado pelo *broker* faz uma ordenação — priorização — prévia, baseado em uma série de características, para definir a ordem das máquinas virtuais controladas pelo *broker* a serem gerenciadas;

- Estimador de Tempo de Conclusão de Máquina Virtual → o algoritmo emprega, opcionalmente, um estimador de tempo restante para a conclusão de uma máquina virtual, baseado em dados históricos do comportamento do usuário que a utiliza, para definir alguns aspectos, como por exemplo se ela deve ser migrada ou não. O emprego dessa estimativa tem o propósito de evitar que a máquina virtual seja finalizada durante a migração, evitando uso desnecessário da rede e potencial ligamento desnecessário de máquinas físicas;

- Independência de Tipo de *Data Center* quanto à Configuração de Máquinas Físicas → o algoritmo pode ser executado tanto em *data centers* homogêneos — com todas

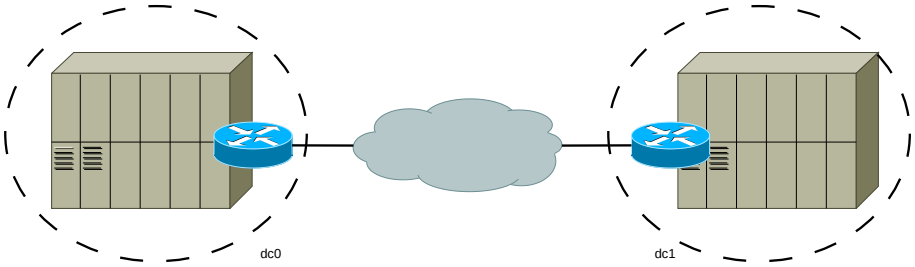

Figura 1.3: Modelo de Geodistribuição de Nuvens trabalhado nesta proposta

as máquinas com especificações idênticas — quanto heterogêneos — com agrupamentos de máquinas com configurações distintas;

- Arbitrariedade do Tamanho de *Data Centers* → o algoritmo pode ser executado em um *data center* de tamanho arbitrário, ou seja, tanto em *data centers* pequenos (ex: aqueles que operam em névoa) quanto *data centers* grandes;

- Uso Racional de *Data Centers* Sem Penalização Severa de Tempos → uma forma ótima de economizar energia em um *data center* que opera em nuvem seria usar somente um servidor, o mais eficiente em energia, para hospedar todas as máquinas virtuais ingressantes. No entanto, essa ação tornaria imensos o *makespan* e o tempo para conclusão de cargas individuais — impraticável. O algoritmo deve ser capaz de utilizar de forma otimizada todos os recursos que ele possui no *data center* de modo a evitar penalizar — e se possível melhorar — os tempos;

- Qualidade de Serviço → se uma máquina virtual de alta prioridade é solicitada, então o algoritmo deve trabalhar para que esta seja atendida — ainda que isso signifique um maior consumo de energia — além de tentar reduzir o tempo de processamento desta;

- Gerenciamento Distribuído de Energia → o algoritmo emprega o desligamento de máquinas físicas e partes da rede desnecessárias;

- Escalonamento Dinâmico de Tensão e Frequência (DVFS) → os servidores podem utilizar tecnologias para ajustar sua tensão e frequência — e consequentemente o consumo de energia — proporcional às cargas que estão processando;

- Migração de Máquinas Virtuais → Quando uma máquina virtual estiver sendo processada em um servidor de baixa eficiência em energia, e existir alguma outra máquina física mais eficiente em energia, então verificar a possibilidade de migrar a máquina virtual em questão.

Nossa contribuição é significante de duas maneiras; (i) em nosso conhecimento, este é o primeiro trabalho a aplicar todos estes princípios fundamentais para o processo de escalonamento de máquinas virtuais na computação em nuvem; (ii) nosso estudo sistemático mostra uma abordagem que traz economia de energia — e consequente redução de potencial poluição gerada — significantes aos *data centers* que compõem nuvens.

O modelo de submissão com o qual os algoritmos propostos trabalham é o *bag-of-tasks*, ou seja, máquinas virtuais ingressantes não têm como pré-requisito a finalização de máquinas virtuais anteriores específicas, embora o algoritmo possa ser adaptado com relativa facilidade para lidar com este cenário.

Portanto, a contribuição deste trabalho pode ser sumarizada como:

- Estudo detalhado dos impactos das redes de alta velocidade em *data centers*, em especial no que diz respeito ao consumo de energia, ao *makespan*, aos tempos de processamento de cargas individuais e ao número de migrações;

- Proposta de um modelo empírico para estimar o consumo de energia em um processo de escalonamento, de modo que, abrangendo um número limitado de dimensões e situações, possibilite uma noção do comportamento de algoritmos de escalonamento e fomente bases para elaboração de outros modelos;

- Proposta de um algoritmo ciente de energia que apresente resultados satisfatórios no processo de escalonamento de máquinas virtuais, trazendo como contribuição para esse processo a ciência da largura de banda, de modo a prover benefícios em cenários constituídos por servidores heterogêneos em largura de banda;

- Proposta de um algoritmo de escalonamento de máquinas virtuais ciente de energia, trazendo como inovação a ciência da topologia da rede, apresentando resultados satisfatórios em um conjunto de possibilidades de configurações de *data centers* — homogêneos, heterogêneos, geodistribuídos, não geodistribuídos, de tamanhos arbitrários, com ou sem SLA, que suportem ou não a migração de máquinas virtuais, com usuários com comportamentos arbitrários ou que possuam uma tendência, com máquinas virtuais de prioridades iguais ou de múltiplas prioridades.

Este trabalho está organizado da seguinte maneira: no Capítulo 2 apresentamos conceitos básicos para balizar o entendimento deste trabalho, a metodologia seguida é apresentada no Capítulo 3, o Capítulo 4 apresenta um estudo detalhado dos impactos das redes de alta velocidade no consumo de energia das nuvens, bem como o aprofundamento deste estudo para contemplar *makespan*, tempo de execução de cargas e número de migrações das nuvens. No Capítulo 5 apresentamos um algoritmo de alocação de máquinas virtuais ciente de energia e de largura de banda com as informações colhidas nos estudos prévios e, em seguida, aprofundamos o escalonamento para considerar topologias de redes no Capítulo 6. No Capítulo 7 apresentamos as conclusões obtidas deste trabalho.

Capítulo 2

Conceitos Básicos e Trabalhos Relacionados

Nesse capítulo são apresentados conceitos básicos necessários para o entendimento deste trabalho, bem como trabalhos que aplicaram tais conceitos ou estão relacionados com este.

2.1 Computação em Nuvem

Knorr et al. [66] definem computação em nuvem como sendo um "modelo de suplemento, consumo e entrega para serviços em Tecnologia da Informação baseado na internet, que tipicamente envolve uma disposição sobre a internet de recursos dinamicamente escaláveis e frequentemente virtualizados". Já Vaquero et al. [119] apresentam nuvens como sendo grandes repositórios de recursos virtualizados (hardware, plataformas de desenvolvimento e/ou serviços), facilmente acessíveis. Estes recursos podem ser reconfigurados dinamicamente de modo a se ajustar de acordo com as cargas, otimizando a utilização destes mesmos recursos. Este repositório de recursos é tipicamente explorado utilizando um modelo do tipo pagamento-por-uso, onde os fornecedores de infraestrutura oferecem garantias no formato de *Service Level Agreements* (SLAs).

A computação em nuvem tem transformado grande parte da indústria de Tecnologia da Informação, tornando softwares mais atrativos como um serviço e modelando a forma como os dispositivos de hardware são projetados e adquiridos. Desenvolvedores com ideias inovadoras para novos serviços de Internet não mais necessitam de gastar seu potencial financeiro adquirindo hardware para implantar seus serviços e nem de um administrador para gerenciá-lo. E a consequência disso é o ininterrupto e crescente interesse nessa área, como podemos ver no gráfico da evolução do interesse dos últimos 10 anos, provido pelo Google Trends e apresentado na Figura 2.1.

Além disso, o custo de utilização de computação em nuvem é menor e o resultado é bem mais rápido, o que a torna escalável. Ambrust et al. [7] vão além e afirmam que a elasticidade de recursos em uma nuvem não tem precedentes na história da tecnologia da informação. A elasticidade se trata de uma das principais características na computação em nuvem, podendo ser entendida como a capacidade do ambiente computacional em

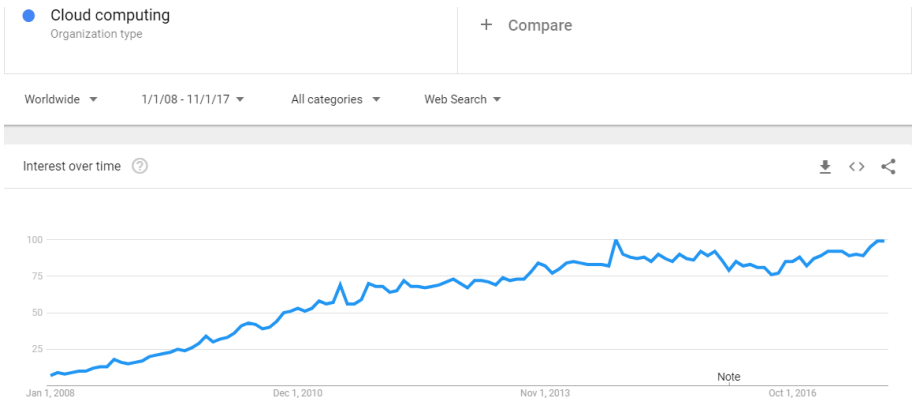

Figura 2.1: Google Trends: Interesse na Computação em Nuvem

nuvem de aumentar ou diminuir os recursos demandados e aprovisionados para cada usuário, aumentando ou diminuindo a capacidade disponibilizada. Quando o usuário necessita de maior capacidade da nuvem, ele solicita essa maior capacidade, e quando tal capacidade não é mais necessária ela é liberada. Essa alocação dinâmica de recursos permite a economia de escala.

Do ponto de vista físico, as nuvens são compostas por um ou mais *data centers* e estes, por sua vez, tipicamente compostos por milhares de máquinas em rede capazes de realizar trabalhos submetidos por usuários. As máquinas que compõem os *data centers* são ligadas entre si.

Normalmente um servidor efetua o escalonamento desses trabalhos e os servidores que os recebem efetuam o processamento desejado.

Do ponto de vista do usuário, as nuvens podem ser acessadas como instâncias de máquinas virtuais. Por exemplo, na plataforma de computação em nuvem Amazon EC2, os usuários podem usar computadores virtuais para suas próprias aplicações, permitindo a execução de um serviço na rede no qual o usuário poderá iniciar uma imagem de uma máquina virtual, a qual é dado o nome de instância.

Este trabalho atua na área de infraestrutura como serviço (IaaS) da computação em nuvem, em especial na seleção de máquinas físicas para hospedar máquinas virtuais.

2.2 Computação Verde

Uma das mais clássicas definições de computação verde é dada por Murugesan [88]: "o estudo e prática de projetar, construir, utilizar e dispôr de computadores, servidores e sistemas associados — como monitores, impressoras, dispositivos de armazenamento, sistemas de redes e comunicação — de forma eficiente e efetiva com impacto mínimo ao meio ambiente".

Michael Dell [32] afirma que sempre acreditou que Tecnologia da Informação fosse o motor de uma economia eficiente, mas ela também pode se direcionar para ser mais verde, enquanto Wang [125] sugere que a ligação entre computação verde e o baixo consumo energético é imediata. Diversas iniciativas têm sido realizadas para tornar a computação menos poluidora no sentido de consumir menos energia e poluir menos o ambiente. Existem

centenas de iniciativas dedicadas em desenvolver a eficiência energética em ecossistemas de *data centers*, tanto nos governos [93] quanto na indústria [46].

Para atingir os objetivos da computação verde, algumas das principais abordagens estão relacionadas com longevidade de produtos, reciclagem de materiais; além dos itens contemplados neste trabalho, sendo eles eficiência de algoritmos, alocação de recursos, virtualização, servidores de terminais e administração de energia.

A virtualização permite atingir os objetivos da computação verde através de consolidação e uso de dispositivos apropriados. Pode-se exemplificar consolidação com o seguinte cenário: no passado era necessário que cada sistema computacional tivesse sua própria estrutura de armazenamento para funcionar. A virtualização em armazenamento permite que sistemas acessem um subsistema de armazenamento que esteja em algum lugar na rede. Isso também quer dizer que cópias de dados armazenados em cada disco de cada sistema agora também podem ser armazenados no mesmo subsistema de armazenamento compartilhado. Está claro que esta abordagem reduz o número de dispositivos de armazenamentos necessários, a quantidade de energia requerida, o calor gerado, e como efeito colateral também reduz os custos administrativos tais como *backups*, manutenção de armazenamentos de arquivos e afins.

Com relação ao uso de dispositivos apropriados, pode-se usar o seguinte cenário: note que em uma organização há softwares e arquivos que são acessados com frequências distintas. Pode-se, por exemplo, dar prioridade para armazenar e executar programas que são usados com maior frequência em virtualizações executadas nas máquinas mais eficientes em energia, enquanto tenta-se manter desligados equipamentos menos eficientes em energia, que poderão ser usados para armazenar softwares e arquivos utilizados esporadicamente em virtualizações de máquinas menos eficientes em energia.

Este trabalho aplica os ideais de computação verde, no sentido de reduzir o consumo de energia consumido por *data centers* que operam em nuvens.

2.3 Escalonamento Dinâmico de Tensão e Frequência

Escalonamento Dinâmico de Tensão (DVS - *Dynamic Voltage Scaling*) [102] é uma técnica bem conhecida em sistemas digitais, especialmente em projetos de microprocessadores de alto desempenho e baixo consumo de energia [61], de gerência de energia em uma arquitetura computacional onde a tensão de determinado componente pode ser alterada de acordo com parâmetros específicos. No âmbito computacional, o aumento de tensão propicia maior estabilidade de um hardware, ao custo de reduzir sua vida útil, por exemplo, com eletromigração ou injeção de transportador quente (HCI); e a redução de tensão permite reduzir o consumo de energia elétrica do dispositivo de hardware.

Em circuitos digitais baseados no transistor MOSFET, é possível efetuar a alternância de estados de tensões de alimentação entre "alta" e "baixa". Fabricantes de processadores se beneficiam dos recursos de DVS para minimizar o consumo de energia de seus dispositivos e, também, reduzem a frequência de tais equipamentos, permitindo-os permanecer estáveis. Esta técnica de escalonar dinamicamente a tensão e a frequência recebe o nome de *DVFS* (*Dynamic Frequency and Voltage Scaling*).

As tecnologias *Intel SpeedStep* [26] e *AMD Coll'n'Quiet* [5] ajustam, automaticamente, em nível de hardware, a frequência e tensão de operação dos seus processadores de acordo com suas cargas de trabalho. Se há altas cargas de trabalho, o processador aumenta os níveis de tensão e frequência, caso contrário diminui estes níveis. Os estados possíveis de frequência e tensão são chamados de *P-states*. Embora os processadores venham com essas tecnologias, o suporte a elas muitas vezes vem desativado na placa mãe, requerendo atenção do administrador do sistema para que as ative.

Para calcular o impacto em energia quando o *DVFS* está ativado, é necessário verificar a diferença entre a energia consumida pelo processador, operando em baixa frequência e tensão, e a energia consumida pelo mesmo processador trabalhando em capacidade máxima. A potência consumida pelo núcleo do processador pode ser obtida a partir da fórmula $P = cfV^2$, onde P é a potência consumida pelo processador em *watts*; c é a capacitância dos transistores internos comutada por ciclo de *clock* do processador e medida em *farads* (determinado pela litografia, constante independentemente da frequência ou tensão); f é a frequência de trabalho do processador em *hertz* e V é a tensão de alimentação da CPU em *volts*. A energia consumida pelo processador pode ser calculada pela potência na qual ele trabalha multiplicada pelo tempo de operação. A consequência disso é que a energia consumida cresce linearmente com a frequência de trabalho do processador e quadraticamente com a tensão.

Para demonstrar a influência das tecnologias supramencionadas no consumo de energia (e consequentemente de energia dissipada), é tomado como base um processador Intel Pentium M de 1,6 *GHz*, com a tecnologia *SpeedStep* ativada. Neste processador as frequências se ajustam de 600 *MHz* até 1,6 *GHz*, em passos de 200 *MHz*. A tensão utilizada na frequência máxima é de 1,484 V e na frequência mínima é 0,956 V. Pode-se verificar que o consumo de energia, no ajuste da frequência máxima para a frequência mínima, é reduzido em aproximadamente 84% e, portanto, bastante significativo. Os cálculos são apresentados na Equação (2.1).

$$1 - \frac{600 \times 0,956^2}{1600 \times 1,484^2} \approx 1 - \frac{548,362}{3523,61} \approx 84\% \tag{2.1}$$

Para melhor visualização, consolidamos a relação entre os *P-states*, tensão, frequência e economia de energia na Figura 2.2.

Neste trabalho aplicamos o conceito de *DVFS* com o objetivo de possuir um consumo de energia inteligente, usando a potência do processador adaptada às cargas que necessita processar.

2.4 Migração de Máquinas Virtuais

A tecnologia de máquinas virtuais é antiga, da década de 1960 [27], tendo apresentado uma grande evolução e se tornado um componente básico para *data centers* e aglomerados, principalmente devido às suas capacidades de isolamento de cargas de trabalho, consolidação e migração [9]. De modo geral esse recurso permite servir múltiplos usuários de uma forma segura, eficiente e flexível. Como resultado, essas infraestruturas virtu-

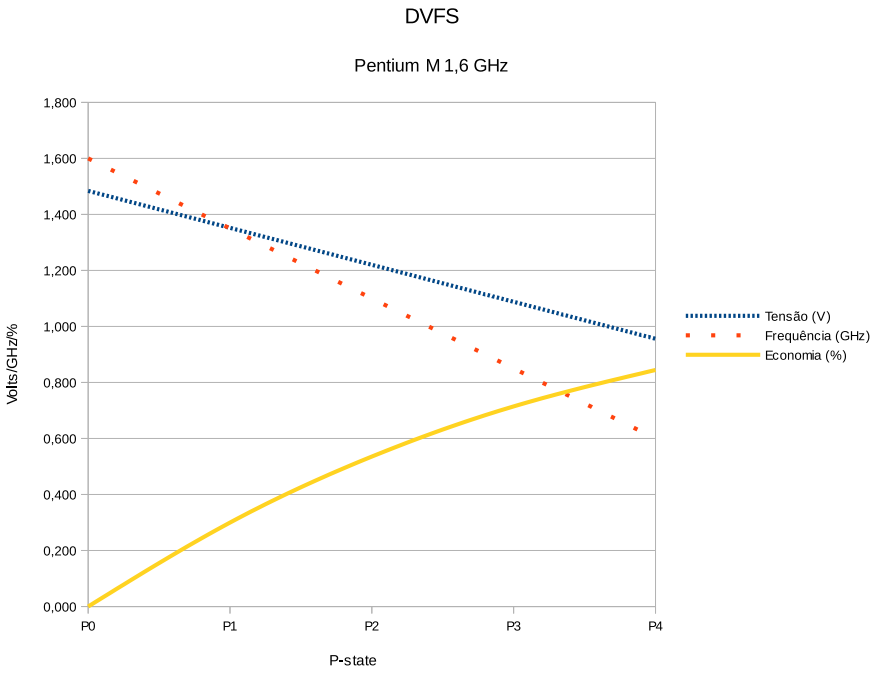

Figura 2.2: DVFS: Relação entre *P-states*, Tensão, Frequência e Economia de Energia

alizadas são consideradas componentes-chave para conduzir o emergente paradigma da Computação em Nuvem [21].

A migração de cargas virtuais possibilita aperfeiçoar a gerência e o desempenho de sistemas. Mais especificamente, as razões que justificam a migração de máquinas virtuais em um sistema de produção incluem: a necessidade de fazer balanceamento de carga, que pode ser conseguida através da migração de máquinas virtuais de servidores sobrecarregados/superaquecidos; e a necessidade de seletivamente desligar servidores para manutenção após migrar suas cargas de trabalho para outros servidores [124]. Essa migração pode ser feita de modo *cold* — desativando a máquina virtual e a migrando entre servidores — ou *hot*, também conhecida como *live* — feita sem gerar indisponibilidade de serviço e é pouco perceptível para o usuário, podendo ser realizada, por exemplo, para retirar a carga de máquinas físicas subutilizadas para desligá-las [24]. Uma das ferramentas que possibilita efetuar esta migração é o vMotion [122]. Notamos também o emprego de migração de máquinas virtuais no contexto de tolerância a falhas, com o intuito de ser um mecanismo para provimento de alta disponibilidade, conforme observamos no extenso estudo de Endo et al. [37].

Neste trabalho empregamos o recurso da migração de máquinas virtuais com o objetivo de reduzir o consumo de energia. Mais especificamente, os algoritmos que apresentamos buscam, quando possível, migrar máquinas virtuais de servidores menos eficientes em energia para servidores mais eficientes em energia e, quando estes servidores menos eficientes estiverem ociosos, desligá-los. Do ponto de vista técnico estes desligamentos podem ser feitos com os servidores enviando uma sinalização ACPI para a placa-mãe [81], e podem ser religados via um pacote *Wake-on-LAN* proveniente do escalonador [17]. Em um computador, a interface de configuração e energia avançada (ACPI) provê um padrão

aberto que possibilita sistemas operacionais descobrirem e configurarem componentes de hardware, de modo a realizar gerenciamento de energia através de operações emitidas pelo *kernel* usando listas de instruções. O *Wake-on-LAN*, por sua vez, é um protocolo padrão da indústria para ligar computadores desligados com acesso à energia através de um pacote específico de rede denominado *magic packet*, que carrega como parte de sua carga útil o endereço físico da interface de rede cujo sistema final deverá ser ligado.

Observamos que em pesquisas recentes, como em [130], a comunicação entre máquinas virtuais passou a receber maior atenção de pesquisadores. Em outras palavras, temos visto abordagens que objetivam deixar máquinas virtuais, que apresentam elevada comunicação entre si, na mesma máquina física, ou pelo menos em máquinas físicas próximas. Nesta proposta, não consideramos este contexto, apesar de considerá-lo para trabalhos futuros.

2.5 Redes de Alta Velocidade e Migração de Máquinas Virtuais

Mostramos na Seção 2.4 que a técnica de migração de máquinas virtuais pode reduzir significativamente o consumo de energia de servidores, em especial através da consolidação de um grande número de máquinas virtuais nos servidores mais eficientes em energia.

No entanto, considerando que um *data center* consiste de um conjunto de máquinas físicas, interconectadas por uma rede, disponíveis para receber/migrar máquinas virtuais, observamos que a migração possui um custo energético associado, dado pela Equação (2.2), onde E_c é a energia total consumida no processo da migração, $E_{s,off}$ é a energia consumida para desligar o servidor de origem da migração, $E_{d,on}$ é a energia consumida para ligar o servidor destino da migração (caso esteja previamente desligado), E_{net} é a energia consumida pela rede durante o processo de migração, $E_{s,migration}$ é a energia consumida durante o tempo da migração pelo servidor de origem da migração e $E_{d,migration}$ é a energia consumida durante o tempo da migração pelo servidor de destino da migração ([101]).

$$E_c = E_{s,off} + E_{d,on} + E_{net} + E_{s,migration} + E_{d,migration} \tag{2.2}$$

Máquinas virtuais migram através da rede dos servidores nas quais elas estão hospedadas e, portanto, o tempo da migração está relacionado à largura de banda destes servidores. Mais especificamente, considere um cenário hipotético onde duas máquinas físicas adjacentes — $h1$ e $h2$ — em um *data center* com uma rede ociosa e uma máquina virtual $vm1$ a migrar entre $h1$ e $h2$. Nós podemos observar que quanto maior for a largura de banda disponível entre $h1$ e $h2$, mais rápido a migração ocorrerá. Neste sentido, o aumento da largura de banda alcançada pela *Ethernet* causa um impacto: se adaptadores de redes mais rápidos são colocados tanto em $h1$ quanto $h2$, e se a rede está adequadamente preparada, menor será o tempo consumido pela migração. Observamos que o fator distância pode também introduzir um atraso de propagação. No entanto, considerando o contexto de migração de máquinas virtuais, que envolve o envio de uma grande quantidade de dados, temos que o atraso de transmissão tende a ser muito grande quando comparado a esse, possibilitando, para muitos casos, a consideração do atraso de propagação como pontual.

Note que os valores de E_{net}, $E_{s,migration}$ e $E_{d,migration}$ estão intimamente ligados com o tempo consumido pela migração, uma vez que a energia é numericamente igual ao tempo multiplicado pela potência consumida ($E = P \times t$). Considerando um cenário de redes de alta velocidade, portanto, estes valores tendem a diminuir, o que significa que é possível realizar um maior número de migrações a um custo potencialmente menor de energia (dependente do consumo da rede).

Nedevschi et al. [90] ressaltam que o consumo de energia pelos equipamentos de redes com o aumento da largura de banda tende a ser maior. Consideramos essa característica neste trabalho, e objetivamos responder, para uma gama de cenários, até onde é admissível o aumento da largura de banda frente ao aumento do consumo de energia em prol da eficiência em energia.

Nossa proposta envolve, portanto, estudar de que maneiras redes de alta velocidade podem impactar o consumo de energia no processo de escalonamento de máquinas virtuais em nuvens, no sentido de reduzir os tempos de migração, permitindo um aumento na eficiência em energia do *data center* como um todo.

2.6 Trabalhos Relacionados

Agrawal e Rao [2] propuseram uma nova formulação e mostraram que o escalonamento ciente de energia é uma generalização do problema de escalonamento com mínimo *makespan*. Eles propõem três algoritmos distintos para o escalonamento ciente de energia, e seu trabalho possibilita o entendimento da compensação entre tempo e energia consumida pelo processo de escalonamento.

Beloglazov e Buyya [11] apresentam uma avaliação de heurísticas para a realocação dinâmica de máquinas virtuais utilizando o recurso de migração *live* de acordo com os requerimentos correntes para o desempenho de processamento. Resultados mostraram que as técnicas propostas possibilitaram economia substancial de energia, enquanto garante qualidade de serviço confiável.

Beloglazov et al. [10] propõem um heurísticas para alocação de recursos para um gerenciamento de *data centers* que operam em nuvens. Nesse trabalho é definido um *framework* arquitetural e princípios para uma computação em nuvem eficiente em energia. Baseado nesta arquitetura, eles apresentam um sistema de provisionamento de recursos e algoritmos de alocação. As heurísticas de alocação propostas proveem recursos dos *data centers* para aplicações clientes de modo a melhorar a economia em energia dos *data centers*, enquanto entregando qualidades de serviço negociadas.

Bergen et al. [12] afirmam que esforços de pesquisa são necessários para relacionar o consumo de energia de hardware com o consumo de energia relacionado à execução de programas. Eles investigam os efeitos das cargas dos processadores no consumo de energia do *data center*, tendo como objetivo a redução do consumo de energia atuando com um escalonamento de tarefas adaptativo e com provisionamento adequado de recursos nas nuvens.

Berl et al. [13] estudam e revisam o uso de métodos e técnicas eficientes em energia no contexto da Computação em Nuvem, em especial no campo de hardware e de infra-

estrutura de redes. O trabalho mostra que a instalação de *plug-ins* específicos e centros de controle de energia para redes de larga escala de software e hardware podem atingir impacto significativo na nuvem, reduzindo a energia consumida pela execução do software e pelo hardware, com o aperfeiçoamento do balanceamento de carga e redução da energia de comunicação e do efeito estufa decorrente das emissões de CO_2.

Bertini et al. [14] buscam reduzir o impacto ambiental em *clusters* de servidores web. Eles modelam o problema selecionando quais servidores devem permanecer ligados e como devem ser seus desempenhos de processamento através de Programação Inteira Mista, combinando suas soluções com teoria de controle. Seus resultados aplicados em *data centers* pequenos, de até 10 máquinas, mostraram redução no consumo de energia de até 40%. Também aplicaram controle de QoS para tentar garantir eficiência em energia e uma boa experiência do usuário.

Binder e Suri [16] propõem um algoritmo probabilístico de alocação e despacho de tarefas que minimiza o número de servidores ativos requeridos, gerenciando o ambiente e conseguindo reduzir o consumo de energia, mas garantindo qualidade de serviço em SLAs.

Deboosere et al. [31] estudam diversos aspectos para otimizar a utilização de recursos e a satisfação de usuários. Através de mecanismos de determinar quantos clientes cada servidor deverá hospedar, considerando-se SLAs, foram capazes de aumentar a utilização de recursos em 29% e de reduzirem o consumo de energia em até 36%.

Duy et al. [36] projetam, implementam e avaliam um algoritmo de escalonamento integrando um preditor de redes neurais para otimizar o consumo de energia em servidores de uma nuvem. Este preditor é construído para prever a carga de trabalho futura baseada em um histórico de demanda. De acordo com a predição, o algoritmo desliga servidores ociosos e os reinicia para minimizar o número de servidores em execução, minimizando desta forma também o uso da energia. Para avaliação foram efetuadas simulações com duas cargas de trabalho e verificado que este modelo é capaz de reduzir em até 46% o consumo de energia.

Ferdaus et al. [39] apresentam relevantes informações e taxonomia detalhada para caracterizar e classificar diversos componentes das técnicas de migração e colocação de máquinas virtuais, bem como elaboram um *survey* e uma análise comparativa das técnicas do estado da arte. Além de elencar vários aspectos e divulgar conhecimentos das estratégias de migração e colocação de máquinas virtuais cientes de rede, e de apresentar algoritmos propostos pela comunidade científica, o *survey* identifica os benefícios e limitações das técnicas existentes e discute as direções da pesquisa futura.

Garg et al. [43] propõem um algoritmo quase ótimo de políticas de escalonamento que exploram a heterogeneidade entre múltiplos *data centers* de um provedor de nuvem. São considerados vários fatores relacionados com energia (tais como custo, taxa de emissão de carbono, carga e eficiência de consumo do processador), enquanto se alterna entre diferentes *data centers* de acordo com sua localização, projeto de arquitetura e sistema de administração. Os autores afirmam que as políticas de escalonamento propostas permitem atingir economias de até 25% em comparação com as políticas atuais.

Hines et al. [49] projetam, implementam e avaliam o mecanismo pós-cópia de migração *live* para máquinas virtuais em redes *gigabit*. Este mecanismo protela a transferência dos conteúdos da memória de uma máquina virtual até que o estado do processador seja

enviado à máquina física de destino; em contraste com a abordagem tradicional, onde primeiro é copiado o estado da memória sobre múltiplas interações seguidas por uma transferência final do estado do processados. É verificado que a estratégia pós-cópia permite reduzir substancialmente o tempo total de migração.

Hu et al. [50] apresentam políticas de escalonamento para redução do consumo de energia em *clusters* que trabalham com máquinas virtuais. Esta abordagem utiliza migração *live* para transferir carga entre os em uma rede *overlay* baseada em anel, e pode reduzir o consumo de energia dos nós do *cluster* como um todo. Medições experimentais mostraram que o método pode reduzir o consumo de energia associado em até 75%.

Kliazovich et al. [62] investigaram atrasos decorrentes de alterações nos modos de energia de máquinas físicas e hardware de rede, bem como suas quantidades para acomodar diferentes padrões de cargas nos *data centers*.

Kliazovich et al. [63] trabalharam com relação às larguras de banda das redes. Eles descobriram que as larguras de banda devem ser levadas em consideração, enquanto para ambientes móveis o escalonamento se torna dependente da alteração de contexto e pode ser atacado por muitas frentes, como eficiência em energia, minimização de custo e maximização de utilização de recursos. Além disso, o papel da malha de comunicação também é enfatizada, e uma solução de escalonamento através de distribuição efetiva de tráfego na rede é apresentado.

Kliazovich et al. [65] consideram o papel do tecido de comunicação no consumo de energia de *data centers* e apresentam uma abordagem de escalonamento ciente de energia e de rede chamada DENS. A metodologia do DENS visa balancear o consumo de energia de um *data center* com o desempenho de cargas de trabalho individuais e as demandas de tráfego; e minimizar o número de servidores ligados e pontos de congestionamento na rede do *data center*.

Kumar e Raghunathan [69] apresentam uma abordagem adaptativa de disposição e consolidação de máquinas virtuais de modo a melhorar a eficiência em energia de um *data center* que opera em nuvem. Nessa abordagem, eles consideram a heterogeneidade entre máquinas físicas e controles térmicos integrados para manter o *data center* em temperatura operacional. A abordagem heurística proposta reduz o consumo de energia com nível aceitável de desempenho.

Kumar et al. [70] analisam técnicas de escalonamento para cenários de computação em nuvem, levando em consideração fatores como a eficiência em energia, minimização de custo e maximização da utilização de recursos. Eles determinam a técnica de escalonamento mais eficiente para um conjunto particular de usuários de acordo com as necessidades e os problemas atacados pelas técnicas estudadas.

Lee et al. [77] propõem um escalonador *multithread* para grades computacionais baseado em algoritmos genéticos com o objetivo de reduzir o consumo de energia, em especial nos horários de pico. Resultados obtidos de um protótipo mostraram que essa implementação pode reduzir a carga de pico substancialmente, além de melhorar os tempos de computação.

Liu et al. [79] estudam a migração *live* de máquinas virtuais em *data centers*, objetivando modelar o desempenho e o consumo de energia. É verificado que os custos de migração variam significativamente para diferentes cargas de trabalho devido à variedade

de configurações e características das cargas. Após analisarem parâmetros chaves que afetam os custos de migração da teoria à prática, eles constroem dois modelos de aplicação para predizer o custo através do uso do conhecimento das cargas de trabalho pelo *hypervisor*, que permite estimar o custo de migração *live* em termos tanto de desempenho quanto de energia. É verificado que as decisões guiadas pelo modelo podem reduzir o custo associado ao consumo de energia das migrações em até aproximadamente 73%.

Luo et al. [99] apresentam um esquema de gerenciamento de recursos focado no modelo IaaS, com excelente desempenho considerando os princípios da computação verde.

Meng et al. [85] propõem usar uma colocação de máquinas virtuais ciente de tráfego com o objetivo de melhorar a escalabilidade. Por otimizar a colocação de máquinas virtuais em máquinas físicas, os padrões de tráfego entre as máquinas virtuais também podem ser melhor alinhados com a distância da comunicação entre elas, por exemplo, máquinas virtuais com um grande uso de largura de banda mútuo podem ser colocadas em máquinas físicas próximas. Um problema de colocação de máquinas virtuais é formulado e sua dificuldade é provada.

Motwani et al. [87] exploram expectativas de SLA e QoS para políticas de consolidação de máquinas virtuais eficientes em energia, e propõem uma política para SLA ciente de energia para consolidação de máquinas virtuais em *data centers* que operam em nuvem.

Nathuji et al. [89] usam máquinas virtuais com o objetivo de reduzir o consumo de energia em ambientes virtualizados em sistemas empresariais.

Piao et al. [98] propõem uma abordagem de colocação e migração de máquinas virtuais objetivando minimizar a transferência de dados e o consumo de tempo. Resultados sugeriram que a abordagem proposta é efetiva em otimizar a transferência de dados entre máquinas virtuais, ajudando a otimizar o desempenho geral da aplicação.

Qian et al. [100] se preocupam com a necessidade de minimização do custo operacional de provedores de *data centers*, apresentando formulações de otimização para minimizar os custos de desgaste e de consumo de energia em diferentes configurações de servidores, em diferentes configurações de *data centers* (entre homogêneos e heterogêneos). São apresentados e analisados dois modelos de agregações — preservando e sem preservar *deadline* para processamento de cargas de trabalho.

Silva e Fonseca [28] propõem um algoritmo de atribuição para grupos de máquinas virtuais ciente de topologia em *data centers*. Este algoritmo foi projetado para ocupar pequenas áreas do *data center* com o objetivo de consolidar os fluxos de rede produzidos pelas máquinas virtuais, sem comprometer a eficiência em energia, e desligando *switches* quando ociosos. Esse algoritmo objetiva a ocupação das menores áreas possíveis da rede em *data centers* não geodistribuídos e, para isso, trabalha com algoritmos auxiliares para localização de subgrafos em topologias Fat Tree — que podem ser modificados para suportar outras topologias.

Srikantaiah et al. [109] estudam como obter a consolidação de eficiência em energia baseando-se no inter-relacionamento entre consumo de energia, utilização de recursos e desempenho das cargas de trabalho consolidadas. É mostrado que existe um ponto ótimo de operação entre estes parâmetros baseado no problema do empacotamento aplicado ao problema de consolidação.

Stage e Setzer [111] trabalham com controle e otimização de largura de banda de algoritmos de migração e elementos de rede relacionados com migração. Eles introduzem modelos de escalonamento de rede ciente de topologia para migrações *live* de máquinas virtuais, propondo um esquema para classificar máquinas virtuais de acordo com suas características de cargas, e propõem modelos de recursos adequados e escalonamento de migrações, levando em consideração requerimentos de largura de banda e da rede para migrações.

Verma et al. [120] investigam projetos, implementações e avaliações de controladores de colocação de aplicações cientes de energia no contexto de ambientes de *clusters* de servidores virtualizados.

Villegas et al. [121] projetam um modelo de serviço em camadas para lidar com federação entre nuvens, em cada nível de camada de serviço (IaaS, PaaS e SaaS), mediada por um *broker* específico para lidar com os problemas neste nível, como SLAs, requisitos de compilação e dados de monitoramento.

Voorsluys et al. [124] avaliam o custo de desempenho da migração *live* de máquinas virtuais em nuvens. É verificado que a migração *live* de máquinas virtuais é capaz de trazer benefícios como melhoria de desempenho, facilidade de gerenciamento e tolerância a falha, enquanto permitindo a movimentação de cargas de trabalho com uma indisponibilidade pequena.No entanto, SLAs podem ser afetados negativamente durante a migração. No entanto, resultados mostram que na maioria dos casos, os custos associados à migração são aceitáveis, mas não devem ser desconsiderados.

Wood et al. [128] estudam e apresentam um sistema que automatiza a monitoração e detecção de pontos sobrecarregados, determinando novos mapeamentos físicos para recursos virtualizados e realizando eventuais migrações necessárias. É mostrado que este sistema é capaz de resolver pontos de sobrecarga em poucas dezenas de segundos, sendo capaz de escalar para grandes *data centers*.

Xu et al. [129] propõem estratégias de escalonamento com múltiplas restrições de QoS para múltiplos *workflows* em computação em nuvem, aumentando a taxa de sucesso para tarefas de múltiplos *workflows* com diferentes requisitos de QoS.

Yanggratoke et al. [131] propõem uma resolução para o problema de alocação de recursos em grandes nuvens baseado em um protocolo que provê uma solução heurística. Como resultado conseguiram minimizar o consumo de energia através de consolidação de servidores quando o sistema está subutilizado e está prevista uma alocação razoável de recursos para o caso de sobrecarga.

Zhang et al. [133] propõem um *framework* escalável para administrar os mapeamentos de máquinas virtuais para máquinas físicas, enquanto resolvem o problema de encontrar o caminho mais eficiente em energia, considerando requerimentos de recursos, entregando resultados bons, tanto e balanceamento de carga quanto em consumo de energia.

Sumarizamos na Tabela 2.1 uma comparação entre os trabalhos os tópicos de interesse de trabalhos que também abordaram a minimização do consumo de energia na computação em nuvem. Nesta tabela elencamos as principais atuações de trabalhos: proposição de novos *Alg*oritmos, abordagem da utilização de *Mig*ração de máquinas virtuais no contexto da computação verde, preocupação com os tempos de *makespan* (Ms) e de execução de cargas individuais (workloads – Wl), ênfase na largura de banda (Bw), atuação em *Data*

*Centers Het*erogêneos, ações no sentido de prover qualidade de serviço cumprindo *SLA*, e abordagem considerando a *Topo*logia de rede dos *data centers*.

Tabela 2.1: Tópicos de interesse de Trabalhos Relacionados

Trabalho	Alg	Mig	Ms	Wl	Bw	Het	SLA	Topo
Agrawal [2]			●					
Beloglazov [11]		●						●
Beloglazov [10]	●	●						
Bergen [12]	●	●						
Berl [13]								●
Bertini [14]							●	
Binder e Suri [16]							●	
Deboosere [31]	●						●	
Duy [36]	●							
Ferdaus [39]	●	●			●			
Garg [43]								
Hines [49]		●			●			
Hu [50]	●	●			●			
Kliazovich [62]	●		●	●			●	
Kliazovich [63]	●			●	●			
Kliazovich et al. [65]	●			●	●		●	●
Kumar [70]					●			
Kumar [69]	●					●		
Lee [77]	●		●	●				
Liu [79]	●	●		●				
Luo [99]	●	●						
Meng [85]		●			●			
Motwani [87]	●	●					●	
Nathuji [89]				●		●		
Piao [98]		●		●				
Qian [100]	●			●		●		
Silva e Fonseca [28]	●	●		●	●			●
Srikantaiah [109]				●				
Stage [111]		●			●			●
Verma [120]		●				●		
Villegas [121]							●	
Voorsluys [124]		●		●			●	
Wood [128]		●						
Xu [129]	●						●	
Yanggratoke [131]				●			●	
Zhang [133]	●	●						
[Este Trabalho]	●	●	●	●	●	●	●	●

Como se pode ver nos trabalhos mencionados, muitas iniciativas foram tomadas em diversos aspectos para tornar os *data centers* que operam numa nuvem mais eficientes em energia e menos poluidoras. Também são vistas iniciativas para lidar com qualidade de serviço em *data centers*. Em trabalhos anteriores [72], nós desenvolvemos e avaliamos um algoritmo para o escalonamento de máquinas virtuais que objetivam minimizar o consumo de energia. Para isso, foram empregadas as técnicas de gerenciamento distribuído de energia (DPM) para desligar servidores inativos, escalonamento dinâmico de tensão e frequência, e a migração de máquinas virtuais. Posteriormente [76], incluímos neste sistema a possibilidade de lidar com prioridades de cargas de trabalhos, possibilitando que o escalonamento conseguisse reduzir substancialmente o tempo de processamento de cargas mais importantes, sem prejudicar criticamente o tempo de processamento de cargas de menores prioridades.

Como continuidade dos trabalhos, e diferentemente dos trabalhos supracitados, no desenvolvimento deste projeto observamos em [73] impactos das emergentes e futuras redes de alta velocidade no consumo de energia de *data centers* que operam em nuvens. Expandimos o trabalho em [74], onde verificamos estes impactos também no contexto de *makespan*, número de migrações e tempos de execuções de cargas de trabalho, inclusive considerando qualidade de serviço. Constatamos que a rápida evolução das especificações de hardware, inclusive de rede, geram uma tendência de *data centers* heterogêneos e, conforme visto no trabalho de Dasgupta et al. [29], a normalização de cargas de trabalho em sistemas heterogêneos, como as nuvens, é um desafio a ser abordado. Abraçamos este desafio, ampliando o escopo do processo de escalonamento anterior. Desenvolvemos em [75] um método de escalonamento de máquinas virtuais ciente de larguras de banda para *data centers*, capaz de prover melhorias na eficiência em energia, *makespan* e tempos de execução de cargas individuais, adequado para *data centers* homogêneos, mas com especial ênfase em *data centers* heterogêneos em largura de banda, abordando também o desafio de normalizar cargas de trabalho em sistemas heterogêneos [29].

Como observado por Zhang et al. [132], existem inúmeros desafios de pesquisa na computação em nuvem. Na evolução dos trabalhos realizados, verificamos que não somente as larguras de banda, mas também as topologias das redes podem fazer diferenças significativas no processo de escalonamento de máquinas virtuais. Em adição às propostas anteriores, também contribuímos para tornar as nuvens mais amigáveis ao meio ambiente, provendo um algoritmo para o escalonamento de máquinas virtuais, visando prover a redução do consumo de energia, aplicando, além das técnicas prévias — DVFS, desligamento de servidores ociosos, ciência de largura de banda, critérios de escolha de máquinas físicas cientes de energia — a ciência da topologia de redes.

Capítulo 3

Metodologia

Desenvolver e validar um algoritmo de escalonamento para computação em nuvem, objetivando minimizar o consumo de energia, levando em consideração aspectos como ciência de energia, ciência de largura de banda, ciência de topologia de redes, uso de recursos de desligamento de servidores ociosos, emprego de migração de máquinas virtuais, DVFS, simultaneamente, não é uma tarefa simples. Consideramos que elaborar tal algoritmo seria uma tarefa muito difícil, não só pela especificidades técnicas do algoritmo, mas pela necessidade de aquisição e desenvolvimento de conhecimento científico necessário, e em especial pela validação.

Diante disso, adotamos a estratégia de dividir para conquistar. Inicialmente, estudamos os impactos das redes de alta velocidade no processo de escalonamento de máquinas virtuais na computação em nuvem, no que diz respeito ao consumo de energia. Depois, expandimos este estudo dos impactos das redes de alta velocidade para contemplar outros fatores importantes, como *makespan*, tempo de execução de cargas individuais e número de migrações. A seguir, desenvolvemos um algoritmo que levasse em consideração a largura de banda da rede. Por fim, desenvolvemos um algoritmo de escalonamento de máquinas virtuais ciente de topologia de rede.

A computação em nuvem normalmente envolve um montante significativo de servidores para o processamento de uma grande quantidade de dados. Por isso, em nível de pesquisa de inovação, não é fácil a implantação imediata de propostas arrojadas em um ambiente físico de produção.

Alguns fatores intrínsecos aos *data centers*, tais como tamanho, incapacidade de desligar máquinas ligadas, necessidade de alterações dinâmicas na estrutura do *data center*, problemas específicos inerentes a testes e pesquisa, necessidade de lidar com um número muito grande de máquinas, além de grandes alterações de infraestrutura em períodos relativamente curtos durante o desenvolvimento dos objetivos propostos, constata-se que a implementação rápida em um ambiente físico real é muito difícil — senão impossível.

Para lidar com estas limitações, relacionadas com à rigidez da infraestrutura, para avaliar o desempenho sistêmico do *data center* sob condições variáveis, observamos comumente o uso de simuladores como ferramentas cruciais para pesquisa. De fato, há vários simuladores disponíveis capazes de lidar com o contexto de nuvens, sendo alguns deles bem aceitos pela comunidade científica. Nós avaliamos uma gama considerável de simuladores cientificamente relevantes para o contexto do nosso trabalho, ou seja, simu-

ladores capazes de lidar com o escalonamento de máquinas virtuais e analisar o consumo de energia. Constatamos, como Malhotra e Jain [82], a existência de um número limitado de simuladores para a computação em nuvem, em especial aqueles capazes de lidar com o consumo de energia. Entre os principais simuladores que avaliamos, citamos o *CloudSim* [22] — e suas extensões WorkflowSim [23], DynamicCloudSim [18], CloudReports [116], CloudMIG Xpress [41], CDOSim [40], FederatedCloudSim [67], CloudAnalyst [127], FTCloudSim [135], RealCloudSim, CloudSimEx, CloudAuction — o *GreenCloud* [64], o GroudSim [96], o CloudExp [58], o iCanCloud [91], o SPECI [110] e o DCSim [117].

Dos simuladores avaliados, escolhemos inicialmente para trabalhar o *CloudSim*, um *toolkit* de simulação baseado em eventos, escrito na linguagem de programação Java, que permite a modelagem e simulação de sistemas computacionais e ambientes de provisionamento para aplicações. Trata-se de um simulador reconhecido academicamente com citações em centenas de trabalhos publicados, que possibilita efetuar análises de duração de simulações e consumo de energia e, por isso, foi escolhido inicialmente como simulador. Para tornar as avaliações possíveis, esse simulador foi estendido com a implementação dos algoritmos *Round-Robin*, *Best Resource Selection — Highest Utilization* e *Lago Allocator*, apresentados na Seção 3.1.

Durante o desenvolvimento do trabalho, no entanto, observamos severas limitações do *CloudSim* e dos outros simuladores avaliados para avaliar uma série de parâmetros, mais detalhados no Capítulo A, em especial relacionados com o consumo de energia pelo núcleo da rede. Por este motivo, desenvolvemos um *toolkit* de simulação, o *SinergyCloud* [25], para possibilitar análises desejáveis para este trabalho.

De modo geral, para todas as etapas de análise do desenvolvimento deste trabalho, adotamos uma sequência de passos bem definida. Primeiramente, definimos e implementamos cenários que permitissem avaliar o objeto de estudo em um simulador, executamos as simulações e analisamos os resultados obtidos. Para poder avaliar os resultados e efetuar comparações, consideramos uma gama de algoritmos de escalonamento apresentados na Seção 3.1, além de um número de configurações, apresentadas na Seção 3.2. Alguns parâmetros de simulação gerais são apresentados na Seção 3.3, enquanto os parâmetros específicos estão apresentados nos capítulos pertinentes.

3.1 Algoritmos Avaliados

Para possibilitar a análise e comparação do processo de escalonamento de máquinas virtuais, com enfoque na economia de energia, precisamos considerar alguns algoritmos de escalonamento de máquinas virtuais em computação em nuvem, que têm a função de encontrar servidores — *Find Host Algorithm* (*FHA*) — para máquinas virtuais. Caso o *broker* — responsável por executar um *FHA* — não encontre um servidor apto a hospedar uma dada máquina virtual ainda não submetida, então a alocação desta é postergada, ou seja, o *broker* tentará em uma próxima atualização, mais tarde, encontrar um servidor para hospedar a máquina virtual em questão. Se o *broker* for capaz de encontrar um servidor válido para hospedar uma máquina virtual já alocada, e este servidor for diferente

do qual hospeda a máquina virtual, então a máquina virtual será migrada para o servidor determinado pelo *FHA*. Empregamos alguns algoritmos bem conhecidos não-cientes de energia, alguns algoritmos cientes de energia e os algoritmos desenvolvidos neste trabalho. Definimos a atualização do *broker* para ocorrer, salvo disposto em contrário, a cada 5 s, ou seja, a cada 5 s o *broker* tenta efetuar a submissão e migração de máquinas virtuais no *data center*. A atualização do *broker* é o evento onde este realiza os seguintes procedimentos: verificação e submissão de máquinas virtuais às máquinas físicas; verificação e submissão de cargas de trabalho às máquinas físicas; verificação e migração de máquinas virtuais instanciadas.

Entre os algoritmos bem conhecidos, podemos citar o *Random*, *Round-Robin* e o *First Available*. Os algoritmos cientes em energia apresentados neste trabalho são o *Bandwidth-Aware Lago Allocator* e o *Topology Energy-Aware Allocator*. Apresentamos os algoritmos usados para comparação que são descritos nas subseções a seguir.

Observamos que, via de regra, os algoritmos cientes de energia tentam consolidar um grande número de máquinas virtuais em um pequeno número de servidores. Heller et al. [48] ressaltam que o resultado de rodar um *data center* com um número mínimo de equipamentos acarreta uma economia de energia, mas os operadores passam a lidar com uma degradação na confiabilidade e no desempenho.

3.1.1 Random

Algoritmo que objetiva encontrar um servidor aleatório para uma máquina virtual.

Operação Este algoritmo monta uma lista de servidores que podem alocar a máquina virtual ingressante, e retorna um elemento aleatório da lista gerada.

Vantagens Este algoritmo tende a fazer um bom balanceamento de carga entre todos os servidores do *data center*.

Desvantagens Quanto ao consumo de energia, este algoritmo usualmente não apresentar resultados bons, uma vez que ele tende a deixar um grande número de servidores ligados.

3.1.2 Round-Robin

Algoritmo de alternância circular para encontrar um servidor para uma máquina virtual.

Operação A primeira máquina virtual ingressante será alocada em um servidor de índice 1; a segunda no servidor de índice 2; e assim sucessivamente, realizando uma alternância circular ao final do processo. Se um servidor é incapaz de hospedar a máquina virtual ingressante, ele será saltado.

Vantagens Este algoritmo tende a fazer um bom balanceamento de carga entre todos os servidores do *data center*, potencialmente melhor que o *Random*.

Desvantagens Quanto ao consumo de energia, este algoritmo usualmente não apresentar resultados bons, uma vez que ele tende a deixar um grande número de servidores ligados.

3.1.3 First Available

Algoritmo que objetiva encontrar o primeiro servidor disponível em uma lista para uma máquina virtual ingressante.

Operação Para cada máquina virtual ingressante, este algoritmo verificará, em uma lista de servidores, em ordem ascendente, qual é o primeiro servidor capaz de hospedar a máquina virtual.

Vantagens Este algoritmo apresenta baixa complexidade, e tende a consolidar um grande número de máquinas virtuais em um número pequeno de servidores — os que apresentam índices mais baixos — economizando energia.

Desvantagens Se os servidores que apresentam número de identificação baixos não são eficientes em energia, o consumo tende a ser grande.

3.1.4 Best Resource Selection — Highest Utilization

Algoritmo que objetiva encontrar o servidor com o nível de utilização mais elevado disponível para uma máquina virtual.

Operação Para cada máquina virtual ingressante, este algoritmo verificará na lista de servidores qual é o que possui a maior utilização — *Highest Utilization* — de acordo com a Equação (3.1), onde *utilization* representa a utilização relativa de um servidor *host*, $host_{mips\_in\_use}$ representa a quantidade de MIPS em uso de *host*, e $host_{mips\_total}$ representa a quantidade total de MIPS que *host* possui. Consideramos a quantidade total de MIPS que um servidor possui como sendo o somatório de MIPS de todos os núcleos de processadores de finalidade geral que este apresenta; por exemplo, se tomarmos um servidor com dois processadores, e cada processador com 4 núcleos de 3.000 MIPS, então a capacidade em MIPS considerada deste servidor será de $2 \times 4 \times 3.000 = 24.000$ MIPS. Um variante deste algoritmo pode ser feito considerando não a utilização, mas a requisição — *Highest Requisition* — de MIPS pelas máquinas virtuais consolidadas, conforme a Equação (3.2), onde *requisition* representa a requisição relativa total de MIPS de um *host* por todas as máquinas virtuais que este hospeda, e $host_{mips\_required\_by\_vms}$ representa a requisição absoluta total de MIPS de um *host* por todas as máquinas virtuais que ele hospeda.

$$utilization = \frac{host_{mips\_in\_use}}{host_{mips\_total}} \tag{3.1}$$

$$requisition = \frac{host_{mips\_required\_by\_vms}}{host_{mips\_total}} \tag{3.2}$$

Vantagens Este algoritmo tenta consolidar um grande número de máquinas virtuais em servidor com maior utilização, potencialmente minimizando o número de migrações e mantendo um maior número de servidores *offline*, tendendo a economizar energia.

Desvantagens Este algoritmo tende a se comportar mal em cargas de trabalho flutuantes; se os primeiros servidores escolhidos não forem eficientes em energia o consumo de energia consolidado do *data center* pode não ser bom; em cenários sem SLAs restritivos em nível de processamento podem impactar significativamente no desempenho das máquinas virtuais alocadas.

3.1.5 Minimum Power Diff

Algoritmo eficiente em energia padrão do *CloudSim* ([19], [22]). Objetiva estimar o consumo de energia após a alocação de uma máquina virtual ingressante para cada servidor, com o objetivo de escolher o servidor que terá menor impacto após a alocação da máquina virtual em questão.

Operação Para cada máquina virtual ingressante, este algoritmo calculará para todos os servidores capazes de hospedar a máquina virtual ingressante a potência estimada após a alocação desta, comparando o resultado obtido com a potência consumida antes da alocação. O servidor que possuir a menor diferença de potência é escolhido.

Vantagens Este algoritmo apresenta bons resultados na minimização do consumo de energia.

Desvantagens Este algoritmo tende a aumentar o *makespan* e tempo de processamento das cargas de trabalho, em especial em cenários onde os servidores com maior poder de processamento possuem menor eficiência em energia.

3.1.6 Lago Allocator

Algoritmo ciente de energia que objetiva selecionar servidores que usando uma sequência de até 4 critérios entre os servidores dos *data centers* [72].

Operação Para cada máquina virtual ingressante, este algoritmo realiza comparações sucessivas, dando prioridade para: (i) o servidor que for mais eficiente em energia; (ii) o servidor que apresenta a menor diferença entre o consumo de energia atual e o consumo de energia estimado após a alocação da máquina virtual; (iii) o servidor que tiver maior utilização; (iv) o servidor que possuir o maior poder de processamento (MIPS).

Vantagens Este algoritmo desempenha bem na minimização do consumo de energia, enquanto minimizando o número de migrações de máquinas virtuais, em especial em *data centers* heterogêneos.

Desvantagens Este algoritmo tende a aumentar o *makespan* e os tempos de processamento das cargas de trabalho, especialmente em cenários onde servidores com maior poder de processamento são menos eficientes em energia.

3.2 Configurações

Cada um dos algoritmos apresentados na subseção anterior pode ser executado com um ou mais recursos disponíveis no *data center*, onde consideramos os mais comuns no contexto de escalonamento ciente de energia. Os algoritmos apresentados foram implementados com diferentes combinações dos seguintes recursos:

- *Power Awareness* → Habilidade de desligar servidores ociosos;
- *DVFS* → Escalonamento Dinâmico de Tensão e Frequência;
- *Migração de Máquinas Virtuais* → Permite que ocorra a migração de máquinas virtuais entre os servidores.

Nas simulações com *DVFS* ativado — ou seja, empregando-se tecnologias vistas em [26] — assume-se que os servidores que estiverem trabalhando no nível mínimo de processamento (ociosos) consomem 70% de sua potência máxima. Em outras palavras, em uma modelagem analítica foi considerado que a potência consumida pelos componentes do servidor, tais como placa mãe, memórias voláteis e permanentes, e processador em operação mínima, será de 70% da potência máxima consumida pelo servidor. É adotado um modelo linear proporcional à carga de trabalho para calcular a potência consumida, de modo que quando um servidor tiver em carga máxima ele consumirá 100% de sua potência. Exemplificamos na Figura 3.1 esta relação entre nível de processamento e potência consumida. Neste exemplo, quando um servidor de 250 W está operando em 0% de sua capacidade ele consome 175 W; quando operando a 50% de sua capacidade ele consome 212 W; e quando operando na capacidade máxima ele consome 250 W; portanto a função que rege o consumo de potência é $P = 175 + 75 \times getCpuUsage(h)$, onde P é a potência consumida por um servidor em *watts* e $getCpuUsage(h)$ retorna o uso percentual da capacidade de processamento do servidor h. Este valor de 70% é corroborado por alguns estudos na área ([38], [71], [51] e [105]).

Para algumas simulações, também consideramos a possibilidade de usar o recurso de qualidade de serviço baseada em prioridades de carga [76], provida por um algoritmo de escalonamento de cargas de trabalho, que atua em paralelo ao algoritmo de escalonamento de máquinas virtuais. Com esse recurso, se suportado pelo algoritmo de escalonamento, cargas de trabalho poderão ser priorizadas, com o objetivo de finalizar as mais importantes em menor tempo.

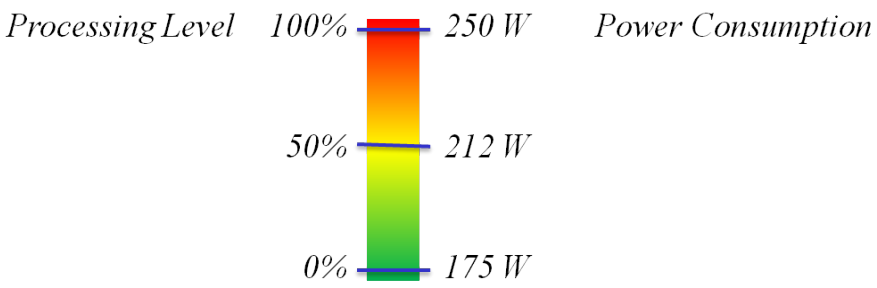

Figura 3.1: Modelo DVFS Linear: Relação entre Processamento e Potência em um Servidor de 250 W com DVFS

3.3 Modelagem das Simulações

Realizamos a validação das propostas através de implementação em simulador e coleta de dados, analisando fatores tais como consumo de energia e tempo, além de impactos causados pelo objetivo de pesquisa em questão. Descrevemos nesta seção os parâmetros comuns considerados para efetuar todas as simulações, salvo disposto em contrário. Parâmetros específicos estão explicitados nos capítulos concernentes.

O desenvolvimento deste trabalho foi alicerçado em alguns princípios definidos por Koomey [68], sendo eles: diversidade de carga, flexibilidade da administração de energia e escolha do local mais eficiente possível.

Um *data center* de pouco uso num contexto de ciência de energia não é um bom cenário para avaliar algoritmos de escalonamento, pois neste caso teríamos a maioria dos servidores desligados. Portanto, focamos em nossa análise quando ocorre maior consumo de potência do *data center*, isto é, em especial, quando existem picos de processamento. Notamos, contudo, que apesar de projetarmos e analisarmos nossas simulações para estes cenários que são significativos para a nossa proposta, em ambientes reais é possível encontrar taxas de utilização dos data centers abaixo de 25% [97]. Para representar este comportamento, consideramos o modelo de distribuição de tarefas *bag-of-tasks single-shot* [56] nos *data centers*, onde todas as cargas a serem processadas são submetidas no início da simulação. Para análise, em cada cenário estudado — para cada tipo de *data center*, para cada tamanho de *data center*, para cada algoritmo, para cada conjunto de configuração possível, etc. — executamos 30 (trinta) simulações e obtivemos as médias e intervalo de confiança em 95% dos valores obtidos. O modelo *bag-of-tasks* referencia um conjunto de tarefas paralelas fracamente acopladas, executadas para produzir um resultado combinado significativo. As submissões *bag-of-tasks* dominam as cargas de trabalho na computação em grade, tanto pelo número de tarefas quanto pelos recursos consumidos, sendo contabilizadas em mais de 75% das tarefas e do tempo consumido pela CPU neste tipo de computação [55].

As métricas que consideramos mais relevantes no desenvolvimento deste trabalho foram o consumo de energia, o *makespan* e o tempo de conclusão das *cloudlets* — cargas de trabalho em uma terminologia do *CloudSim* — considerando suas prioridades. Baseado em [72], e realizando adaptações necessárias, nós consideramos os seguintes critérios para comparar os algoritmos nos cenários estudados: seja A um algoritmo de escalonamento

para computação em nuvem (e.g. HU), C um conjunto de configurações com as quais o algoritmo atua (e.g. desligamento de servidores ociosos, DVFS e migração de máquinas virtuais), e P um conjunto de parâmetros do *data center* (e.g. 10 servidores, 60 máquinas virtuais e 60 *cloudlets*).

- A_1 com configuração C_1 e parâmetros P é considerado melhor do que A_2 com configurações C_2 e os mesmos parâmetros P se o consumo de energia ($(A, C, P)_{pow\_av}$) considerando o intervalo de confiança de 95% em energia ($(A, C, P)_{pow\_ci}$) satisfaz a Inequação (3.3);
- A_1 com configuração C e parâmetros P será inaceitavelmente mais lento do que A_2 com a mesma configuração C e os mesmos parâmetros P se o *makespan* ($(A, C, P)_{mk\_av}$) considerando o intervalo de confiança de 95% em energia ($(A, C, P)_{mk\_ci}$) satisfaz as Inequações (3.4) e (3.5).

$$(A_1, C_1, P)_{pow\_av} + (A_1, C_1, P)_{pow\_ci} < (A_2, C_2, P)_{pow\_av} - (A_2, C_2, P)_{pow\_ci} \tag{3.3}$$

$$(A_1, C, P)_{mk\_av} - (A_1, C, P)_{mk\_ci} > 2 \times ((A_2, C, P)_{mk\_av} - (A_2, C, P)_{mk\_ci}) \tag{3.4}$$

$$(A_1, C, P)_{mk\_av} + (A_1, C, P)_{mk\_ci} > 2 \times ((A_2, C, P)_{mk\_av} + (A_2, C, P)_{mk\_ci}) \tag{3.5}$$

Em outras palavras, para cada cenário, o algoritmo a ser considerado o melhor é aquele que satisfaz a Inequação (3.3) e que as Inequações (3.4) e (3.5) não sejam verdadeiras para cada conjunto de possíveis configurações e parâmetros; caso contrário (se nenhum algoritmo atende estes itens) então não é possível dizer que um algoritmo é melhor que outro.

Quanto ao tipo de migração de máquinas virtuais, consideramos o uso da migração *live*, com valor de degradação de desempenho das máquinas virtuais em migração definido em 10% (padrão do *CloudSim*), i.e., máquinas virtuais terão um processamento efetivo de 90% no processamento de suas cargas durante a migração.

Quanto ao tamanho dos *data centers*, inicialmente consideramos três tamanhos: $s10$ (10 servidores), $s100$ (100 servidores) e $s1000$ (1.000 servidores). Estas definições de tamanho foram baseadas nos conceitos de tamanhos de *data center* de [59], respectivamente, pequeno, médio e grande.

Consideramos que cada máquina virtual recebe uma carga de trabalho com um número de milhões de instruções para processamento e, após o processamento desta carga, a máquina virtual é finalizada. Esta estratégia foi adotada para (i) que não ocorra a degradação de desempenho do processamento de cargas de trabalho pelas máquinas virtuais, (ii) evitar adicionar um grau de complexidade desnecessário ao escopo deste trabalho e

(iii) melhor avaliar os casos de grande processamento de cargas de *data centers*, onde os algoritmos podem ser melhor avaliados para desempenharem seus papeis.

Os algoritmos desenvolvidos neste trabalho visam atuar em cenários de *data centers* com configurações arbitrárias. Neste sentido, objetivamos que as análises se dessem tanto em *data centers* de pelo menos dois tipos, (i) homogêneos — nos quais todos os servidores possuem configurações de hardware idênticas — e (ii) heterogêneos — com agrupamentos de servidores de configurações distintas, por exemplo, em 10 servidores divididos em 3 agrupamentos poderão ser encontrados um agrupamento de 4 servidores com configuração c_1, um agrupamento de 3 servidores com configuração c_2 e um agrupamento de 3 servidores com configuração c_3. Deste modo, se especificarmos que temos um *data center* $s100$ homogêneo, queremos dizer que estamos tratando de um *data center* constituído por 100 máquinas com configurações de hardware idênticas.

Em cima destes *data centers* realizamos baterias de simulações com diversos cenários, considerando servidores com configurações de hardware recentes. Nos *data centers* heterogêneos, o critério para a seleção de processadores foi tentar criar uma disparidade significativa e capacidade de processamento entre os servidores, para verificar em uma situação real como os algoritmos se comportam em situações de heterogeneidade grande, mas em *data centers* que usam processadores de última geração. Para efeitos de simulação consideramos que cada operação realizada pelas cargas de trabalho pode ser concretizada em um ciclo de *clock* dos processadores, ou seja, há uma razão de 1:1 entre frequência (MHz) e MIPS.

O principal foco deste trabalho é a minimização do consumo de energia. No entanto, devemos evitar a escassez de recursos de hardware, de modo que os resultados das simulações não sejam comprometidos por problemas que não são alvo dos algoritmos. Por outro lado, servidores com configurações irrealistas não devem ser usados. Realizadas estas considerações, é definido que as configurações comuns para servidores de *data centers* homogêneos e heterogêneos são:

- Entidades de Processamento → Cada servidor tem um processador, podendo este ter múltiplos núcleos;
- Espaço em Disco → Cada servidor possui 1 TB de espaço em disco;
- Arquitetura dos servidores do *Data Center* → É assumido que todos os servidores usam arquitetura baseada em x64;
- Sistema Operacional → O sistema operacional escolhido para os servidores do *data center* foi GNU/Linux;
- Monitor de Máquina Virtual → O monitor de máquina virtual escolhido para os servidores do *data center* foi o Xen;
- Tamanho de Arquivo das *Cloudlets* → Cada *cloudlet* submetida ao *data center*, antes do processamento, possui 300 bytes (padrão dos modelos do *CloudSim*);
- Tamanho da Saída das *Cloudlets* → Cada *cloudlet* possui 300 bytes de dados após o processamento (padrão dos modelos do *CloudSim*);

- Monitor de Máquina Virtual → O monitor de máquina virtual, para as máquinas virtuais, é o Xen;

Foram utilizadas as configurações padrões dos simuladores usados para os parâmetros de simulação não explicitamente reportados neste trabalho.

Quanto às métricas apresentadas neste trabalho usamos: W para potência; kWh para energia; segundos para tempo; milhões de instruções por segundo (MIPS) para velocidade de processamento; milhões de instruções (MI) para tamanhos de cargas.

Consideramos nas simulações que envolvem ciência de energia que todos os servidores e comutadores de pacotes intermediários iniciam desligados, e que a simulação estará concluída após o processamento de todas as cargas.

Consideramos que os servidores e *switches* serão desligados quando ociosos. Consideramos ociosidade no contexto de servidores quando eles não estiverem processando e nem em uma transmissão de máquinas virtuais — uma aplicação de [123] a este contexto.

Nas simulações as quais não especificamos a topologia de rede utilizada, consideramos a topologia *single-hop star*, na qual existe um *switch* conectado a todos os sistemas finais.

Existe uma métrica comumente usada na indústria denominada *Power Usage Effectiveness* (PUE) [1], utilizada para o cálculo de eficiência em uso de energia. O PUE para um *data center* pode ser calculado de acordo com a fórmula apresentada em (3.6).

$$PUE = \frac{TFP}{ITEP} \tag{3.6}$$

Nessa fórmula, TFP (*Total Facility Power*) é a potência total consumida pelo *data center* (incluindo instalações e equipamentos) e $ITEP$ (*IT Equipment Power*) é a potência consumida pelos equipamentos de TI. Além disso existem categorias nas quais o PUE se encaixa, dependendo do tipo proveniente da sua fonte energética. Por exemplo, se a categoria de cálculo do PUE for 0, isto implica que os *data centers* que serão avaliados são totalmente elétricos, não consumindo outros tipos de energia (como gás natural, etc). Esta é uma métrica que se aplica normalmente a ambientes reais, tendo em vista que o ambiente de simulação não considera o consumo de energia das instalações, do tipo de resfriamento, entre outras características de consumo de energia de *data centers*. O cálculo de PUE, portanto, não se aplica a este trabalho, devido a dependência de características de infraestrutura de *data centers* não contempladas em seu escopo.

Com relação ao SLA, nós consideramos as seguintes condições: se uma máquina virtual é alocada, então ela terá 100% de disponibilidade até sua desalocação; não existe garantia de desempenho de processamento mínimo; a política de alocação de máquina virtual é de melhor esforço, no sentido que quando o *broker* receber uma solicitação de máquina virtual ele a tentará alocar imediatamente; e não há garantia de largura mínima de banda para as máquinas virtuais alocadas. Para todas as simulações que realizamos neste trabalho, não observamos violações destes SLAs.

3.4 Lista de Siglas e Acrônimos de Algoritmos, Recursos e Unidades

Para facilitar a leitura deste trabalho, usamos algumas siglas e acrônimos para especificar algoritmos, configurações sob as quais estes atuam, dados estatísticos e unidades de medidas. Listamos tais siglas e acrônimos na Tabela 3.1.

Tabela 3.1: Siglas e Acrônimos de Algoritmos, Recursos e Unidades

Sigla/Acrônimo	Significado
	Algoritmos de Escalonamento
`RND`	*Random*
`RR`	*Round-Robin*
`FA`	*First Available*
`HU`	*Best Resource Selection — Highest Utilization*
`HR`	*Best Resource Selection — Highest Requisition*
`MPD`	*Minimum Power Diff*
`LA`	*Lago Allocator*
`BALA`	*Bandwidth-Aware Lago Allocator*
`TEA`	*Topology Energy-Aware Allocator*
	Recursos Usáveis por Algoritmos de Escalonamento
`N`	Ausência de Gerenciamento Distribuído de Energia (***N**ot Power-Aware*)
`P`	Desligamento de Servidores Ociosos (***P**ower-Aware*)
`D`	Uso de Escalonamento Dinâmico de Tensão e Frequência (***D**VFS*)
`M`	Uso de Migração de Máquinas Virtuais (***M**igration*)
	Estatística e Unidades de Medida
`AV`	Média de Consumo de Energia (***A**verage Energy **C**onsumption*)
`CI`	Intervalo de Confiança de 95% (*95% **C**onfidence **I**nterval*)
MIPS	Milhões de Instruções por Segundo
MI	Milhões de Instruções

Recapitulando, um algoritmo pode ser executado com uma combinação de recursos. Podemos designar, por exemplo, `MPD-PM` como a execução do algoritmo `MPD` executado com os recursos de desligamento de servidores ociosos e de migração de máquinas virtuais.

Capítulo 4

Impactos das Redes de Alta Velocidade

Em nossos estudos observamos que as redes de alta velocidade podem impactar o processo do escalonamento de várias maneiras, em especial no consumo de energia, *makespan*, tempo de conclusão de máquinas virtuais, além do número de migrações que ocorrem nos *data centers*. Nas seções a seguir, apresentamos os estudos que realizamos sobre estes aspectos das referidas redes.

4.1 Impactos das Redes de Alta Velocidade no Consumo de Energia das Nuvens

Redes de alta velocidade impactam a comunicação em *data centers* que operam em nuvem. Esta comunicação é utilizada, notavelmente, no processo de submissão e migração de máquinas virtuais. A submissão e migração de máquinas virtuais são realizadas por algoritmos de escalonamento. Para verificarmos esse impacto, portanto, devemos analisar as consequências de taxas de transmissão maiores nestes algoritmos, bem como de que maneira a ciência de largura de banda pode trazer benefícios à eficiência em energia no processo de escalonamento de máquinas virtuais. Para esta análise, consideramos quatro algoritmos de escalonamento, `RR`, `HU`, `MPD` e `LA`. Além disso, vamos considerar a combinação de recursos — dentre `N`, `P`, `D` e `M` — que cada algoritmo pode utilizar. Para esta análise, utilizamos o simulador *CloudSim*.

Assumimos que, para todas as simulações, o consumo de energia pela rede é fixo, com o objetivo de calcular o quanto as velocidades das redes impactam no consumo de energia, de modo a permitir calcular em quanto o consumo de energia pode subir com as velocidades das redes de modo a manter o *data center* eficiente em energia.

Estamos particularmente interessados na diferença do consumo de energia entre as redes de diferentes velocidades. Seja Ec_{ns_1} a energia consumida pelas máquinas físicas operando com velocidade de rede ns_1, e Ec_{ns_2} a energia consumida pelas máquinas físicas operando com velocidade de rede ns_2, então a diferença máxima de energia aceitável $MAED$ (***M**aximum **A**cceptable **E**nergy **D**ifference*) será dada pela Equação (4.1).

$$MAED = Ec_{ns_1} - Ec_{ns_2} \tag{4.1}$$

Quanto maior é o valor de $MAED$, melhor é a eficiência em energia alcançada pelas redes de velocidades maiores. Por exemplo, se um servidor operando com velocidade de rede de 100 Mbps consumir 200 kWh; e quando operando a 10 Gbps consumir 150 kWh; então, do ponto de vista de eficiência em energia, é aceitável que o consumo da rede poderia subir em até $200 - 150 = 50$ kWh — em virtude da rede mais rápida — quando comparado à rede mais lenta.

4.1.1 Simulações e Análises

Nós executamos simulações para as seguintes taxas de transmissão: 100 Mbps, 1 Gbps, 10 Gbps, 100 Gbps, 1 Tbps e 10 Tbps, e coletamos os resultados para analisar os impactos das velocidades das redes no consumo de energia.

Parâmetros de Simulação

A Tabela 4.1 inclui os parâmetros que utilizamos para a simulação. Campos multivalorados, como tamanhos de cargas de trabalho e MIPS de máquinas virtuais, utilizam distribuição *round-robin*. Por exemplo, a executada de uma *cloudlet* de 20.000 MI em uma VM de 3.000 MIPS consome aproximadamente 7 segundos, representando um processamento relativamente leve; enquanto uma *cloudlet* com 12.500.000 MI em uma máquina virtual com 750 MIPS consome cerca de 5 horas, representando um processamento relativamente pesado. Quanto à distribuição *round-robin*, em uma simulação com 21 máquinas virtuais, 7 delas serão geradas com 750 MIPS, 7 com 1.500 MIPS, e outras 7 com 3.000 MIPS. Nas simulações nós também calculamos o *makespan*.

Tabela 4.1: Parâmetros de Simulação

Parâmetro	Valores
Velocidades de Redes (Gbps)	0.1, 1, 10, 100, 1.000, 10.000
Tamanhos de Cargas de Trabalho (MI)	20.000, 100.000, 500.000, 2.500.000, 12.500.000
Número de Máquinas Virtuais	6 vezes o número de máquinas físicas
Razão Cargas de Trabalho : Máquinas Virtuais	1:1
RAM de Máquinas Físicas	24 GiB
VM MIPS	750, 1.500 e 3.000
Largura de Banda de Máquina Virtual	2,5% da largura de banda total do servidor hospedeiro
VM RAM	128 MiB [34]
Razão VM RAM : RAM do Servidor	1:192

Para este estudo, as *cloudlets* apresentam distribuição *round-robin*, para possibilitar uma variabilidade de processamento generalizada. Além disso, assumimos como seis vezes o número de máquinas virtuais em relação ao número de servidores.

Em *data centers* homogêneos, cada servidor possui consumo de 250 W e um processador baseado no Intel Xeon X5667. Em outras palavras, um processador com 4 núcleos operando a 3.066 MIPS.

Em *data centers* heterogêneos, cada servidor possui um processador, em uma distribuição *round-robin*, baseado em um dos seguintes modelos: Intel Xeon E5603, Intel Xeon E7520, Intel Xeon X5667, AMD Opteron 6204 e AMD Opteron 6220; possuindo, respectivamente, 4 núcleos com 1.600 MIPS cada, 4 núcleos com 1.860 MIPS cada, 4 núcleos com 3.066 MIPS cada, 4 núcleos com 3.300 MIPS cada e 8 núcleos com 3.000 MIPS cada. Além disso, cada servidor apresenta uma das seguintes potências, também distribuídas em *round-robin*: 200, 250, 300 e 350 W. A Tabela 4.2 ilustra como o MIPS/potência dos servidores é distribuído neste tipo de *data center*.

Tabela 4.2: Configuração de Servidores em um *Data Center* Heterogêneo

Servidor	h_0	h_1	h_2	h_3	h_4
MIPS	1.600×4	1.860×4	3.066×4	3.300×4	3.000×8
Potência (W)	200	250	300	350	200

Resultados da Simulação e Análises

Executamos um total de 1.152 simulações, e obtivemos uma margem de erro pequena para o intervalo de confiança considerado, citando por exemplo a margem de erro de 0,0049 para o caso do `LA-PDM` para o caso de um *data center* $s1000$ e heterogêneo com intervalo de confiança de 95%. Nas tabelas apresentadas nesta subseção, representamos por `ER` a redução relativa do consumo de energia do cenário com a rede mais lenta para o cenário com a rede mais rápida.

Apresentamos na Tabela 4.3 as diferenças nos consumos de energia para *data centers* homogêneos, e na Tabela 4.4 estes valores para *data centers* heterogêneos; em ambos os casos para todos os tamanhos, para todas as velocidades e para os quatro algoritmos de escalonamento. Por exemplo, o consumo de energia médio do `RR` em um *data center* $s10$ e homogêneo com rede operando a 10 Tbps é 7,38 W, resultado de $11,63 - 4,25$.

Como é possível observar em todos os casos, o aumento da largura de banda possibilitou a redução no consumo de energia. Isso implica que se considerarmos aumentar a largura de banda do *data center*, de modo que este aumento impacte em um valor inferior ao percentual `ER`, o *data center* será mais eficiente em energia. Além disso, considerando todas as larguras de banda, o algoritmo mais eficiente para *data centers* $s1000$ e homogêneos, $s100$ e heterogêneos, e $s1000$ e heterogêneos, é o `LA`. Isso provavelmente ocorre devido ao fato que o `LA` já era melhor que os outros algoritmos em consumo de energia. Analisando o princípio de redução do número de migrações de máquinas virtuais do `LA`, perceptível especialmente em *data centers* heterogêneos, nós podemos inferir que o consumo de energia é mais afetado pela seleção dos servidores para hospedar uma VM do que a velocidade da rede no processo de escalonamento, mas esta velocidade não pode ser ignorada.

Em *data centers* homogêneos, nós observamos que o tempo de *makespan* é reduzido, em especial em *data centers* $s1000$. Este redução varia de 11,65% (`MPD-PDM`) até 30,09% (`MPD-PM`). Portanto, podemos afirmar que o *makespan* é positivamente impactado com o aumento da velocidade das redes, em especial em *data centers* $s1000$.

Tabela 4.3: Consumo de Energia em *Data Centers* Homogêneos

	AV±CI				ER			
Conf.	RR	HU	MPD	LA	RR	HU	MPD	LA
Data Center $s10$ Homogêneo @ 100 Mbps								
PM	16,21±0,04	5,77±0,24	4,85±0,32	5,42±0,19				
PDM	11,63±0,02	4,28±0,18	5,08±0,09	4,60±0,08				
Data Center $s10$ Homogêneo — Diferença entre 100 Mbps e 10 Tbps								
PM	6,02±0,03	3,07±0,22	2,17±0,30	2,71±0,17	37%	53%	45%	50%
PDM	4,25±0,00	2,07±0,17	2,88±0,07	2,38±0,06	37%	48%	57%	52%
Data Center $s100$ Homogêneo @ 100 Mbps								
PM	164,09±0,06	23,40±0,22	20,62±0,07	21,75±0,11				
PDM	118,04±0,05	19,63±0,14	31,28±0,40	20,67±0,16				
Data Center $s100$ Homogêneo — Diferença entre 100 Mbps e 10 Tbps								
PM	61,34±0,01	9,13±0,21	6,35±0,06	7,44±0,10	37%	39%	31%	34%
PDM	43,67±0,02	6,47±0,13	18,12±0,40	7,48±0,15	37%	33%	58%	36%
Data Center $s1000$ Homogêneo @ 100 Mbps								
PM	1.644,83±0,27	225,76±0,45	199,65±0,33	213,87±0,35				
PDM	1.183,27±0,20	189,90±0,39	326,72±2,03	188,02±0,37				
Data Center $s1000$ Homogêneo — Diferença entre 100 Mbps e 10 Tbps								
PM	616,65±0,12	71,08±0,44	44,92±0,31	60,67±0,33	37%	31%	22%	28%
PDM	439,17±0,10	49,83±0,38	186,66±2,02	49,03±0,36	37%	26%	57%	26%

Com respeito a *data centers* heterogêneos, o *makespan* se comportou de forma diferente. Em *data centers* $s10$, ele variou de uma redução no tempo de 6,44% (`MPD-PDM`) a uma piora de 12,29% (`HU-PDM`); nos *data centers* de tamanho $s100$ houve uma melhoria de 30,06% (`MPD-PM`) a uma piora de 11,99% (`HU-PDM`); e em *data centers* $s1000$ houve uma melhora, em todos os casos, variando de 0,31% (`RR-PDM`) a 34,84% (`MPD-PM`). Portanto, não é possível conclusivamente afirmar que o aumento na velocidade das redes de computadores implica em uma redução no *makespan*, exceto no caso de *data centers* $s1000$.

Com relação ao número de migrações, quando comparamos as taxas de transmissão de 100 Mbps às de 10 Tbps para os algoritmos `HU`, `MPD` e `LA`, podemos efetuar as seguintes observações: (i) em *data centers* homogêneos ocorreu o aumento deste valor, variando de 21,88% (`MPD-PDM` heterogêneo) a 462,72% (`HU-PDM` heterogêneo); (ii) em *data centers* $s100$ as estes valores aumentaram entre 1.369,63% (`MPD-PDM` heterogêneo) e 3.969% (`LA-PDM` homogêneo); e (iii) no caso dos *data centers* $s1000$ este aumento variou entre 2.691,08% (`MPD-PDM` heterogêneo) e 5.238% (`LA-PDM` homogêneo). No caso do `RR`, o aumento — esperado, devido à natureza da alternância do algoritmo — no número de migrações ultrapassou 7.000%.

Tabela 4.4: Consumo de Energia em *Data Centers* Heterogêneos

	AV±CI				ER			
Conf.	RR	HU	MPD	LA	RR	HU	MPD	LA
Data Center $s10$ Heterogêneo @ 100 Mbps								
PM	17,23±0,05	6,41±0,35	4,90±0,25	2,13±0,01				
PDM	12,43±0,03	4,66±0,24	3,13±0,19	1,64±0,01				
Data Center $s10$ Heterogêneo — Diferença entre 100 Mbps e 10 Tbps								
PM	6,43±0,03	3,49±0,29	2,37±0,20	0,22±0,00	37%	54%	48%	10%
PDM	4,55±0,02	2,12±0,19	1,66±0,18	0,17±0,00	37%	45%	53%	10%
Data Center $s100$ Heterogêneo @ 100 Mbps								
PM	180,79±0,06	25,27±0,67	21,38±0,34	12,99±0,07				
PDM	130,83±0,05	21,08±0,70	13,79±0,12	11,58±0,09				
Data Center $s100$ Heterogêneo — Diferença entre 100 Mbps e 10 Tbps								
PM	67,84±0,02	8,15±0,59	10,06±0,26	6,10±0,03	38%	32%	47%	47%
PDM	48,59±0,02	5,30±0,64	7,13±0,09	5,39±0,07	37%	25%	52%	46%
Data Center $s1000$ Heterogêneo @ 100 Mbps								
PM	1.813,52±0,25	232,57±1,80	205,16±0,88	128,16±0,27				
PDM	1.310,22±0,20	190,99±1,64	129,64±0,94	106,96±0,17				
Data Center $s1000$ Heterogêneo — Diferença entre 100 Mbps e 10 Tbps								
PM	683,21±0,12	64,07±1,78	83,41±0,87	52,54±0,25	38%	28%	41%	41%
PDM	487,08±0,10	38,21±1,62	55,85±0,93	40,07±0,16	37%	20%	43%	37%

4.1.2 Um Modelo Empírico para o Consumo de Energia

Baseados nos resultados das simulações, nosso próximo objetivo é determinar um relacionamento empírico entre os diversos fatores considerados neste estudo. No entanto, desenvolver um modelo empírico para estimar o consumo de energia em qualquer *data center*, considerando parâmetros como o número de servidores, número de máquinas virtuais, número de cargas de trabalho, tamanho das cargas de trabalho, largura de banda, potência consumida por cada servidor, modelo de distribuição de cargas e recursos utilizados por cada algoritmo de escalonamento não foi possível após tentarmos inúmeros modelos de regressão. De fato, se considerarmos estas variáveis chegamos a um problema de pelo menos oito dimensões. Decorrente disso, optamos por fixar alguns destes parâmetros para melhor desenvolver uma estimativa do consumo de energia.

Observamos que o número de máquinas virtuais é igual ao número de cargas de trabalho, e este é seis vezes o número de servidores. Além disso, vamos considerar somente as configurações `PM` e `PDM`, que são aquelas que envolvem migração de máquinas virtuais. Com estas suposições, podemos reduzir o problema a três dimensões: número de servidores, velocidade das redes e energia consumida pelo *data center* no modelo de escalonamento estudado.

Após diversos experimentos, nós determinamos funções bidimensionais usando modelos de regressão de ajuste de curva para cada tamanho e tipo de *data centers*. Ou seja, por

esta abordagem conseguimos uma função que recebe como parâmetro de entrada a largura de banda e retorna o consumo de energia estimado pelo *data center*, desconsiderando o equipamento de rede.

Através destas experimentações nós encontramos o seguinte modelo empírico, representado na Equação (4.2), baseado em uma função de decaimento exponencial, como sendo a mais adequada para nosso trabalho.

$$y = y_0 + Ae^{\frac{-x}{t}} \tag{4.2}$$

Baseado na função apresentada, nós verificamos os valores para as constantes y_0, A e t, de modo a permitir que essa função receba o parâmetro x, que representa a velocidade da rede medida em Mbps, e retorna y, que representa o consumo de energia estimado em kWh para os cenários simulados. Portanto, nós usamos esta função como base para estimar o consumo de energia usando todos os algoritmos apresentados neste trabalho com configurações cientes de energia e com o recurso de migração de máquinas virtuais ativado (`PM` e `PDM`).

Para *data centers* homogêneos, nós apresentamos na Tabela 4.5 os valores constantes que foram determinados para serem usados com (4.2) para calcular o consumo de energia estimado. Por exemplo, o consumo de energia de um *data center* $s10$ com configuração `PM` e executando o algoritmo de escalonamento `LA` para os cenários propostos, é bem representado pela equação $y = 2,70282 + 2,83572e^{\frac{-x}{2.784,09551}}$.

Para *data centers* heterogêneos, nós apresentamos na Tabela 4.6 os valores das constantes que devem ser usados na Equação (4.2) para calcular o consumo de energia estimado.

Nós ilustramos nas Figuras 4.1 e 4.2, em escala logarítmica, o comportamento para duas instâncias representativas; aqui o eixo x representa a largura de banda (em Mbps) e o eixo y representa o consumo estimado de energia. O comportamento geral entre os algoritmos é similar, tendo como principal diferença o ponto de estabilização após o qual o aumento da largura de banda não promove impacto significante na redução do consumo de energia.

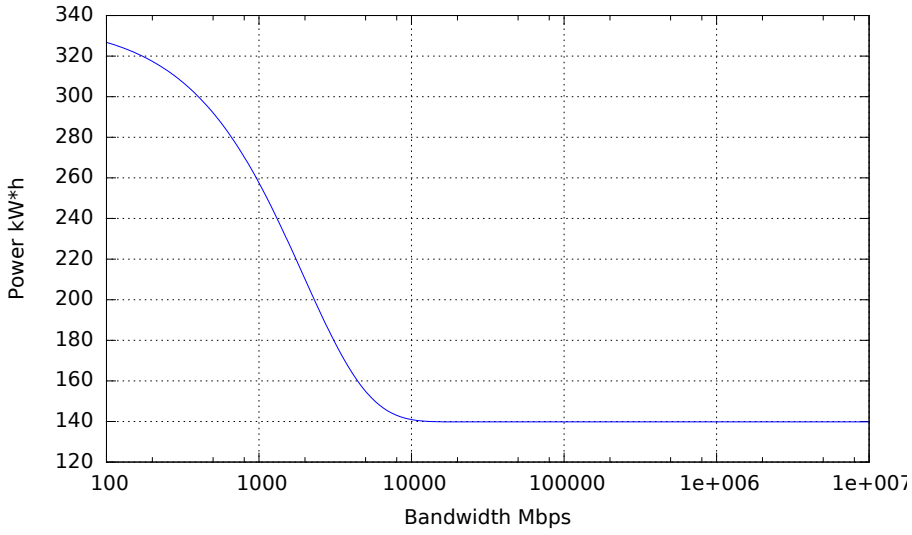

Figura 4.1: Consumo de Energia: 1.000 Servidores, DC Homogêneo, `MPD-PDM`

Tabela 4.5: Constantes da Função de Estimativa de Consumo de Energia para *Data Centers* Homogêneos

Servidores	Algo	Config.	y_0	A	t
10	RR	PM	10,11377	6,4921	3.627,60717
10	HU	PM	2,67814	3,22316	2.636,35684
10	MPD	PM	2,66591	2,27678	2.922,30737
10	LA	PM	2,70282	2,83572	2.784,09551
10	RR	PDM	7,31861	4,60005	3.664,03229
10	HU	PDM	2,18845	2,18644	2.765,98852
10	MPD	PDM	2,18891	3,01694	2.537,66161
10	LA	PDM	2,20451	2,49302	2.696,06922
100	RR	PM	101,97131	65,75713	3.494,18772
100	HU	PM	14,26654	9,95405	1.164,8013
100	MPD	PM	14,2691	6,94683	1.119,5355
100	LA	PM	14,32023	8,08427	1.183,21755
100	RR	PDM	73,81804	46,80913	3.498,05655
100	HU	PDM	13,16057	7,06886	1.137,12121
100	MPD	PDM	13,0925	18,99625	2.418,08483
100	LA	PDM	13,18211	8,14598	1.196,60522
1.000	RR	PM	1.019,40599	668,72109	3.718,6519
1.000	HU	PM	154,68796	80,15318	832,0401
1.000	MPD	PM	154,72357	51,68791	713,64577
1.000	LA	PM	153,20288	68,36449	837,00897
1.000	RR	PDM	737,88314	476,20822	3.718,2470
1.000	HU	PDM	140,05495	56,44313	804,32979
1.000	MPD	PDM	139,78004	196,85604	1.946,46312
1.000	LA	PDM	138,99904	55,67264	785,98756

Como é possível observar nos gráficos, existe um limite superior de largura de banda que impacta o consumo de energia, após o qual virtualmente nenhuma mudança é notada. Considerando potências de 10, nós temos um limite superior de 1 Gbps para *data centers* heterogêneos $s100$ e $s1000$ para o algoritmo `HU-*` (`HU` com qualquer configuração); um limite superior de 10 Gbps ocorre para *data centers* homogêneos $s100$ e $s1000$ para os algoritmos `HU-*`, `MPD-PM` e `LA-*`; bem como para *data centers* heterogêneos $s10$ (`HU-PM` e `LA-*`), $s100$ (`MPD-PM` e `LA-PM`) e $s1000$ (`LA-*`). Para todos os demais casos, o limite superior foi de 100 Gbps — em nenhum caso ocorreu um limite superior a 100 Gbps. Concluindo, como mostramos na Figura 4.3, para os cenários avaliados, em 8,3% dos casos o limite superior foi de 1 Gbps, em 35,4% dos casos o limite superior foi de 10 Gbps, e em 56,2% dos casos o limite superior foi de 100 Gbps. Em *data centers* maiores, o benefício decorrente do aumento da largura de banda tendeu a ser menor.

Com as funções apresentadas, e para os cenários estudados, é possível determinar até qual limite o aumento do consumo de energia é aceitável com o objetivo de manter o *data center* eficiente em energia. Por exemplo, considere calcular o impacto em energia quando aumentando a velocidade da rede de um *data center* $s100$ e heterogêneo de 1 Gbps

Tabela 4.6: Constantes da Função de Estimativa de Consumo de Energia para *Data Centers* Heterogêneos

Servidores	Algo	Config.	y_0	A	t
10	RR	PM	10,70612	6,93592	3.614,48551
10	HU	PM	2,90955	3,75625	1.422,85952
10	MPD	PM	2,51583	2,48945	2.551,2765
10	LA	PM	1,91525	0,29118	335,26806
10	RR	PDM	7,80419	4,92109	3.632,86573
10	HU	PDM	2,4976	2,26853	2.454,85047
10	MPD	PDM	1,47001	1,74756	1.893,08258
10	LA	PDM	1,47167	0,2211	346,46182
100	RR	PM	112,15479	72,56471	3.471,04423
100	HU	PM	16,86513	51,5428	55,14381
100	MPD	PM	11,3056	10,78597	1.456,71931
100	LA	PM	6,87437	6,65421	1.194,87392
100	RR	PDM	81,66981	51,95114	3.454,01466
100	HU	PDM	15,2402	37,39807	53,83431
100	MPD	PDM	6,64382	7,49041	2.203,77764
100	LA	PDM	6,18258	5,7997	1.391,19357
1.000	RR	PM	1.120,8683	738,12808	3.650,6713
1.000	HU	PM	165,82083	411,50103	54,97927
1.000	MPD	PM	121,68817	88,60645	1.676,17552
1.000	LA	PM	75,62853	59,07129	852,78524
1.000	RR	PDM	816,42356	526,49848	3.663,93983
1.000	HU	PDM	148,34036	273,31252	53,83228
1.000	MPD	PDM	73,22041	59,25821	3.261,3144
1.000	LA	PDM	66,89072	44,69028	915,50623

para 10 Gbps, no qual o algoritmo de escalonamento `MPD-PDM` é utilizado. Substituindo os parâmetros y_0, A e t da Equação (4.2) pelos valores fornecidos na Tabela 4.6, nós obtemos um resultado de 11,40 kWh (um bom resultado — pela simulação o valor calculado foi de 11,41 kWh); e substituindo os mesmos parâmetros pelos valores referentes à velocidade de 10 Gbps nós obtemos como resultado 6,72 kWh (também um bom resultado – o valor calculado por simulação foi de 6,66 kWh). Seguindo a Equação (4.1), nós subtraímos os valores de estimativa calculados, tendo como resultado 4,68 kWh, que representa o quanto o limite máximo de quanto o consumo de energia pode subir se a rede de 10 Gbps for implantada. Por exemplo, se a melhoria na rede implicar em um incremento de 10% no consumo de energia, o resultado será $6,72 + 6,72 \times 0,1 = 7,39$ kWh, abaixo do consumo originalmente estimado (11,40 kWh), o que pode ser interpretado como: "a implantação de uma taxa de 10 Gbps para este caso provê eficiência em energia".

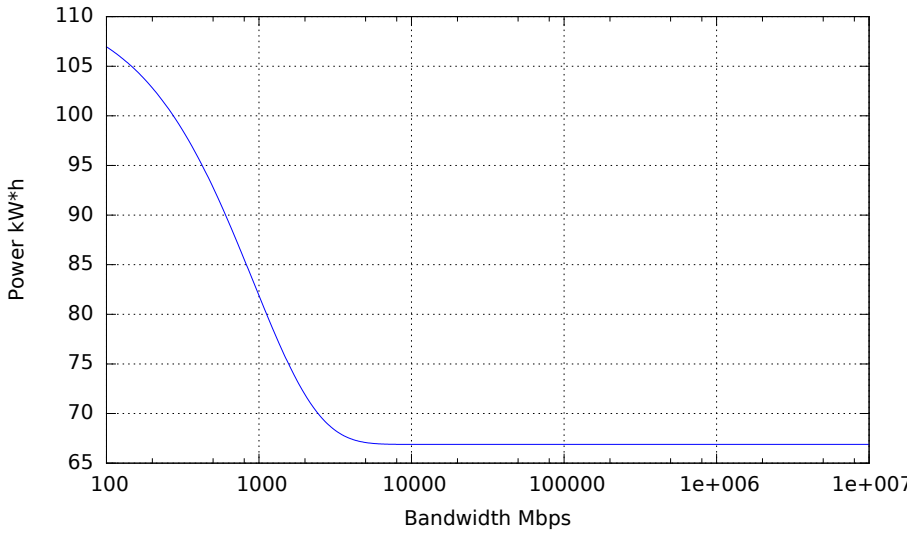

Figura 4.2: Consumo de Energia: 1.000 Servidores, DC Heterogêneo, `LA-PDM`

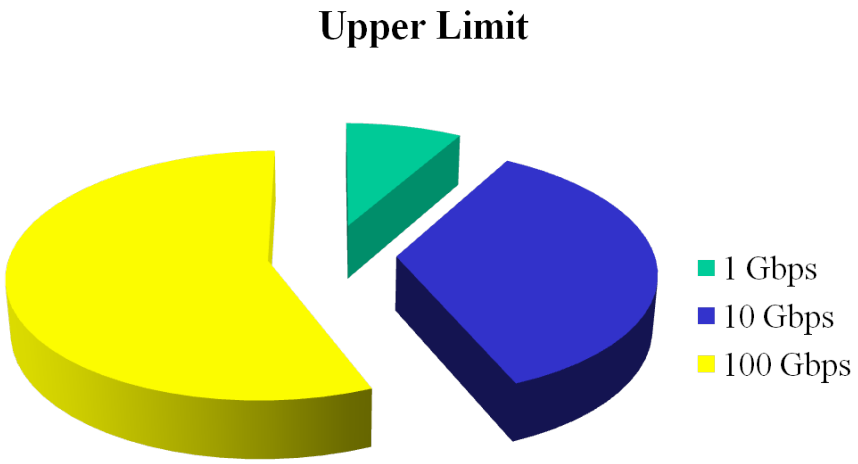

Figura 4.3: Limite Superior de Ganhos Potenciais Providos por Redes de Alta Velocidade

4.2 Impactos das Redes de Alta Velocidade no *Makespan*, Tempo de Execução de Cargas e Número de Migrações das Nuvens

Nesta seção continuamos os estudos sobre os impactos das redes de alta velocidade nas nuvens, mas desta vez nos voltamos para o *makespan*, tempo de execução de cargas e número de migrações de máquinas virtuais. Para este estudo, selecionamos três algoritmos de escalonamento de máquinas virtuais representativos: `HU`, `MPD` e `LA`.

Para isso, consideramos dois algoritmos de submissão de cargas de trabalho: um algoritmo ordinário — sem QoS — e outro algoritmo que permite que cargas com altas

prioridades sejam processadas preferencialmente, para ser executado para cada um dos algoritmos de escalonamento de máquinas virtuais e para cada uma das configurações cientes de energia. Neste caso, como cargas com altas prioridades podem ser submetidas mais rapidamente, elas tendem a ser finalizadas mais cedo. Como objetivamos avaliar os tempos de execução de cargas de trabalho, optamos por fazer uma análise estendida, envolvendo qualidade de serviço.

Consideramos as possíveis configurações para os algoritmos de escalonamento de máquinas virtuais como `P`, `D` e `M`. Além disso, levamos em conta que os algoritmos de escalonamento de máquinas virtuais podem estar trabalhando com um escalonador ordinário de cargas de trabalho (`-`) ou com um escalonador com QoS baseado em prioridade de cargas (`Q`). Por exemplo, referimos a `LA-PDMQ` como sendo o algoritmo de escalonamento de máquinas virtuais o *Lago Allocator*, executando com a habilidade de desligar máquinas físicas ociosas, em um *data center* onde todas as máquinas físicas estão com o recurso de DVFS ativado, com a habilidade de realizar a migração de máquinas virtuais, e utilizando um escalonador de cargas de trabalho com QoS.

Nós conduzimos uma série de simulações para estudar de forma abrangente a migração de máquinas virtuais e o escalonamento com QoS baseado em prioridade de carga, e os impactos no *makespan*, migração e tempos de execução. Para isso, estendemos o *CloudSim* implementando o algoritmo de escalonamento de cargas de trabalho com QoS baseado em prioridade de cargas, e a interação deste aos algoritmos de escalonamento de máquinas virtuais implementados no *CloudSim*.

Nós realizamos simulações para cada algoritmo de escalonamento de máquinas virtuais, com todas as configurações possíveis, e para cada largura de banda selecionada para o *data center*: 1 Gbps, 10 Gbps, 100 Gbps, 1 Tbps e 10 Tbps, e para cada tamanho de *data center*.

4.2.1 Configuração do *Data Center*

O número de máquinas virtuais que o *data center* processará é seis vezes o número de máquinas físicas, e cada máquina virtual receberá uma *cloudlet*. Estes números foram encontrados através de experimentação, e bem apropriados devido ao fato que eles são suficientemente grandes para mostrar o comportamento do *data center* sob cargas de alto processamento, mas não grandes demais para desorientar as simulações realizadas. Por exemplo, um *data center* com 100 máquinas físicas receberá 600 máquinas virtuais e, portanto, 600 *cloudlets*. As configurações comuns para máquinas físicas, em *data centers* homogêneos e heterogêneos, usados nos nossos estudos, são:

- RAM → Cada servidor possui 8 GiB de RAM;
- MIPS de Máquinas Virtuais → As máquinas virtuais do *data center* apresentam capacidades de processamento de 1.000 MIPS, 2.000 MIPS e 3.000 MIPS, em uma distribuição *round-robin*. Por exemplo, em uma simulação com 20 máquinas virtuais haverá 7 máquinas virtuais com 1.000 MIPS, 7 máquinas virtuais com 2.000 MIPS e 6 máquinas virtuais com 3.000 MIPS;

- Tamanho das *Cloudlets* → As *cloudlets* no *data center* possuem 30.000, 120.000, 480.000, 1.920.000, 7.680.000 milhões de instruções em uma distribuição *round-robin*. Estes tamanhos foram escolhidos para dar uma variabilidade generalizada de processamento. Por exemplo, uma *cloudlet* com 30.000 milhões de instruções em uma máquina virtual com 3.000 MIPS consome aproximadamente 10 segundos para ser processada, representando um processamento relativamente leve, enquanto uma *cloudlet* de 7.680.000 milhões de instruções em uma máquina virtual de 1.000 MIPS consome cerca de 2 horas, representando um processamento relativamente pesado.

- Prioridade de *Cloudlets* → Cada *cloudlet* submetida ao *data center* tem uma prioridade da seguinte lista, em distribuição *round-robin*: *alta*, *normal*, *normal*, *baixa*. Em outras palavras, 50% das cargas de trabalho recebem prioridade *normal*, enquanto 25% das *cloudlets* possuem prioridade *alta*, e 25% das *cloudlets* apresentam prioridade *baixa*;

- Largura de Banda de Máquina Virtual → Cada máquina virtual recebe 8,33% da largura de banda total do servidor que a processa.

4.2.2 Configurações para *Data Centers* Homogêneos e Heterogêneos

Para as simulações executadas em *data centers* homogêneos, as seguintes configurações foram adotadas:

- MIPS dos Servidores → Cada servidor em um *data center* homogêneo tem um processador baseado no Intel Xeon X5667; em outras palavras, um processador de 4 núcleos operando cada um a 3,06 GHz e, portanto, de acordo com as considerações de simulação, 3.066 MIPS por núcleo;

- Potência dos Servidores → A potência consumida para cada servidor em um *data center* homogêneo é 250 W;

- RAM por Máquina Virtual → Cada máquina virtual possui 1 GiB de RAM.

Para as simulações executadas em *data centers* heterogêneos, as seguintes configurações foram adotadas:

- MIPS dos Servidores → Cada servidor em um *data center* heterogêneo possui um processador, em distribuição *round-robin*, baseado em um dos seguintes modelos: Intel Xeon E5603, Intel Xeon E7520, Intel Xeon X5667, AMD Opteron 6204, AMD Opteron 6220; tendo, respectivamente, 4 núcleos com 1.600 MIPS cada, 4 núcleos com 1.860 MIPS cada, 4 núcleos com 3.066 MIPS cada, 4 núcleos com 3.300 MIPS cada e 8 núcleos com 3.000 MIPS cada;

- Potência dos Servidores → Cada servidor em um *data center* heterogêneo possui uma das seguintes potências, em distribuição *round-robin*: 200, 250, 300 e 350 W.

- RAM por Máquina Virtual → Cada máquina virtual em um *data center* heterogêneo possui a seguinte quantidade de RAM, em distribuição *round-robin*: 512 e 1.536 MiB.

Ilustramos na Tabela 4.7 como a relação MIPS/Potência é distribuída neste tipo de *data center*.

Tabela 4.7: Configuração de Servidores em um *Data Center* Heterogêneo

Servidor	h_0	h_1	h_2	h_3	h_4	h_5
MIPS×Núcleos	1.600×4	1.860×4	3.066×4	3.300×4	3.000×8	1.600×4
Potência (W)	200	250	300	350	200	250

4.2.3 Resultados: *Makespan*

Nós agora apresentamos e analisamos os resultados do *makespan* para os casos estudados.

Apresentamos nas Figuras 4.4–4.9 o *makespan* para os algoritmos de escalonamento de máquinas virtuais `HU`, `MPD` e `LA`, sob as configurações `PM`, `PDM`, `PMQ` e `PDMQ` para *data centers* homogêneos e duas taxas de transmissão, de 1 Gbps e 100 Gbps. De modo similar, para *data centers* heterogêneos, os resultados são apresentados nas Figuras 4.10–4.15.

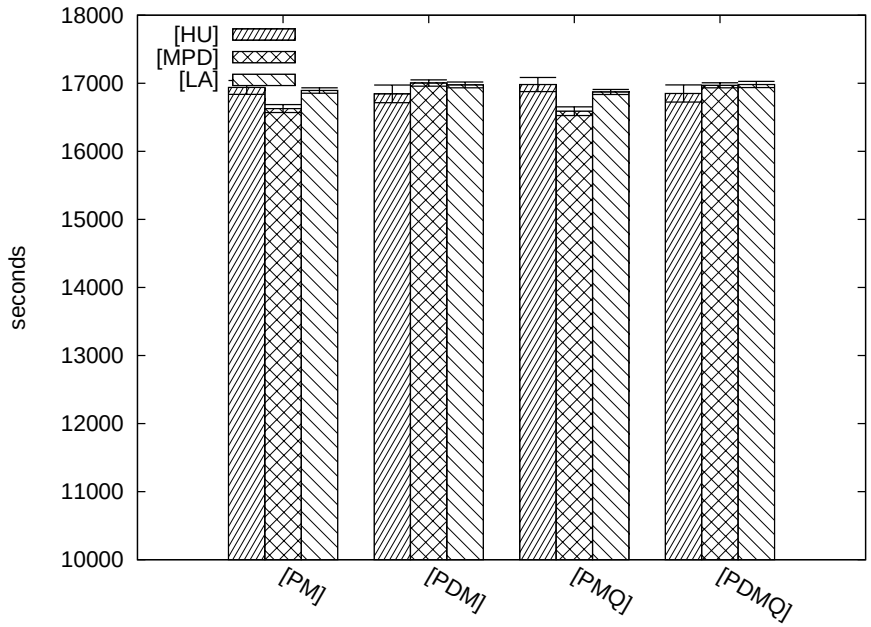

Figura 4.4: *Makespan*: *Data Center* Homogêneo — 10 Servidores @ 1 Gbps

Nós observamos que o aumento na velocidade das redes leva a uma redução no *makespan*. Isso pode ser explicado pelo fato que velocidades mais rápidas de redes possibilitam a migração de máquinas virtuais em tempo inferior ao das redes mais lentas, portanto, reduzindo o potencial impacto negativo de desempenho durante a migração. Como resultado, passa a existir maior eficiência para as máquinas virtuais processarem suas cargas

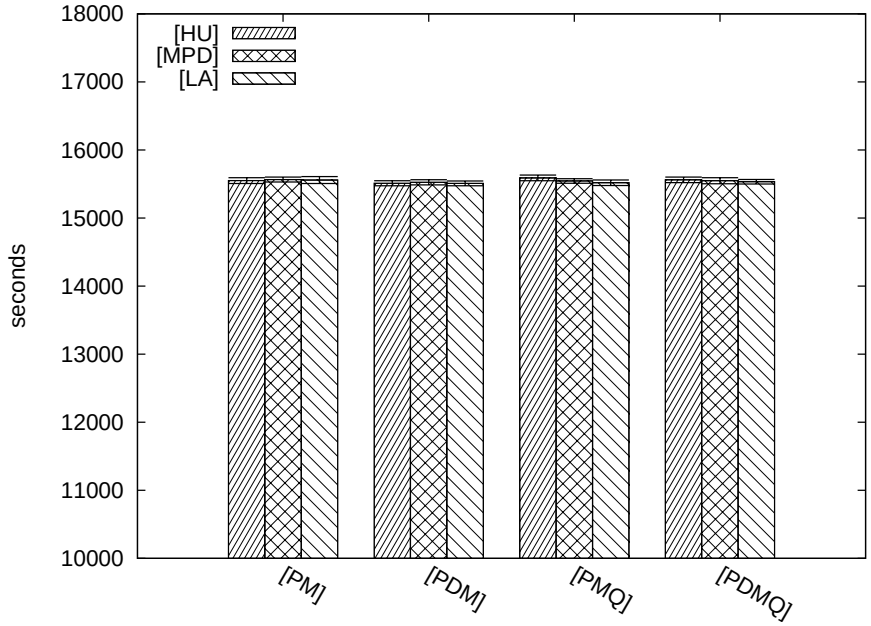

Figura 4.5: *Makespan*: *Data Center* Homogêneo — 10 Servidores @ 100 Gbps

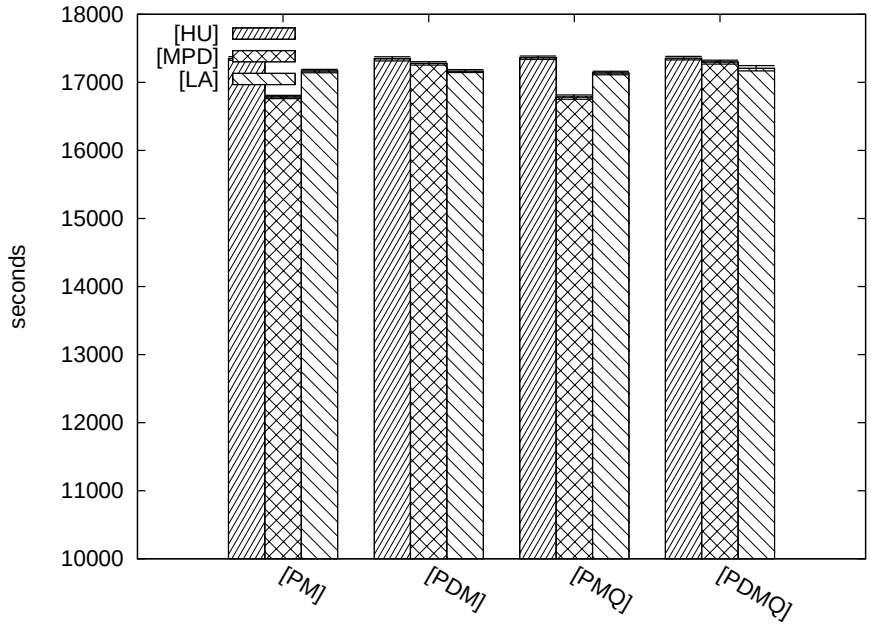

Figura 4.6: *Makespan*: *Data Center* Homogêneo — 100 Servidores @ 1 Gbps

de trabalho. Mais do que isso, ainda que considerássemos que as migrações das máquinas virtuais não reduzissem o desempenho das máquinas virtuais, ainda assim o *makespan* poderia ser reduzido, pois as velocidades mais rápidas da rede permitiram que o *broker* submetesse as tarefas mais rapidamente aos servidores, possibilitando um processamento inicial mais cedo das tarefas existentes.

Um importante ponto para se notar aqui é que o aumento das velocidades das redes beneficia mais alguns algoritmos de escalonamento do que outros. Analisando a estrutura de algoritmos de escalonamentos, nós notamos que uma estratégia comum em algoritmos

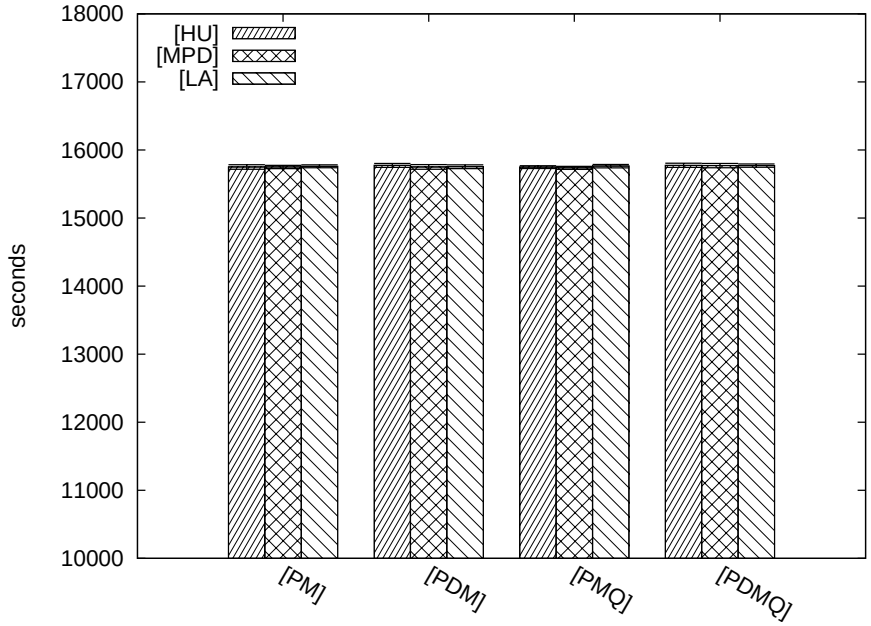

Figura 4.7: *Makespan*: *Data Center* Homogêneo — 100 Servidores @ 100 Gbps

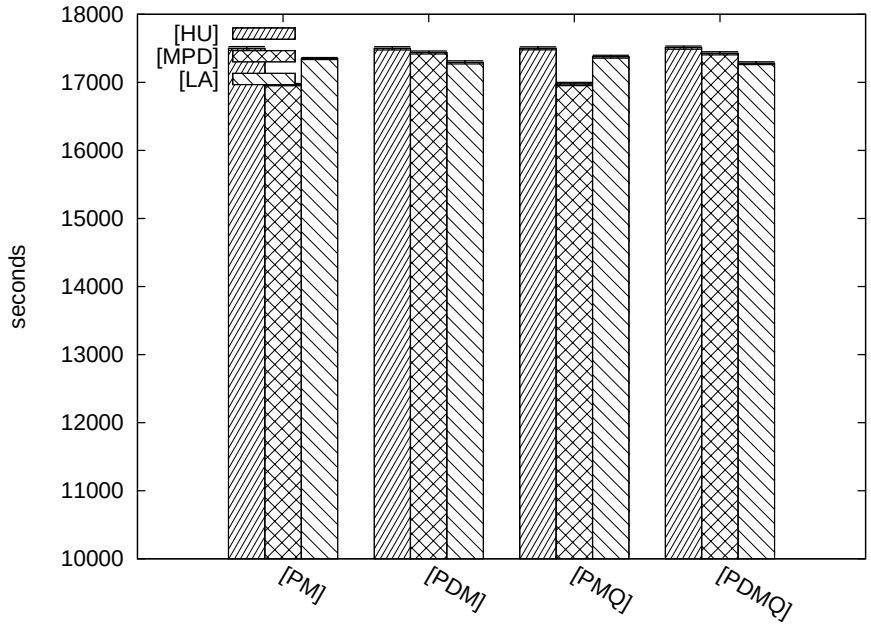

Figura 4.8: *Makespan*: *Data Center* Homogêneo — 1.000 Servidores @ 1 Gbps

eficientes em energia é reduzir o número de migrações, tentando dispôr máquinas virtuais em servidores nos quais elas tendam a permanecer por mais tempo sem a necessidade de migração. Isso ocorre como mostrado na Subseção 4.2.5. Portanto, se o aumento da largura de banda nas redes impacta a taxa de migrações, e alguns algoritmos intencionalmente tendem a realizar menor número de migrações, então é esperado que estes se beneficiem menos do que aqueles que realizam um grande número de migrações.

A seguir, nós consideramos o impacto do *makespan* como uma função do aumento da velocidade da rede de 1 Gbps para 10 Tbps; conforme se observa nas Figuras 4.16 e 4.17

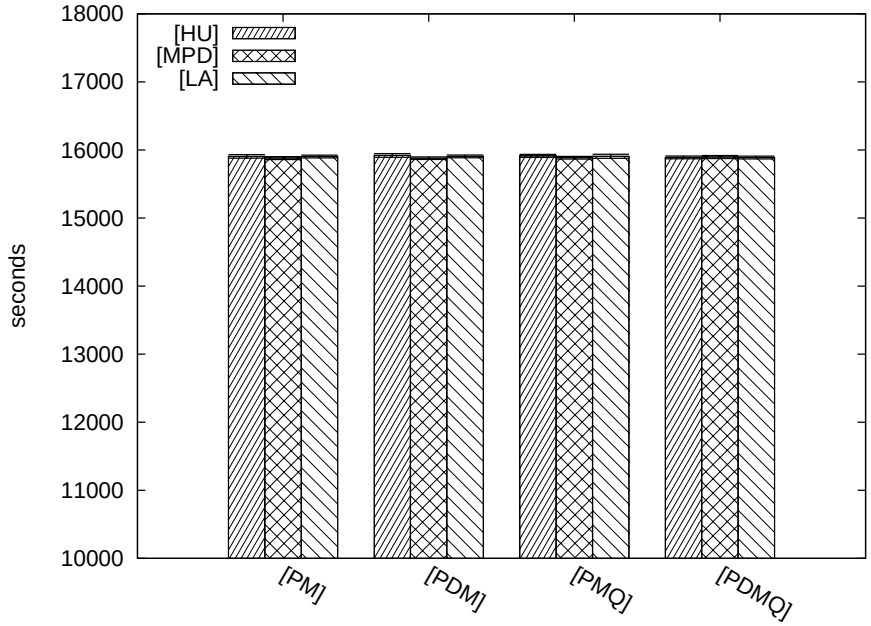

Figura 4.9: *Makespan*: *Data Center* Homogêneo — 1.000 Servidores @ 100 Gbps

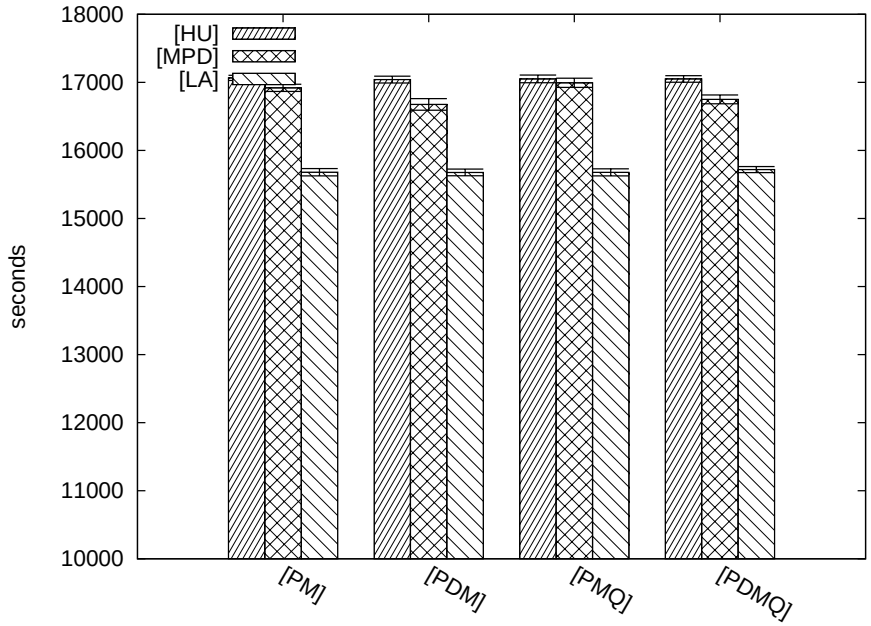

Figura 4.10: *Makespan*: *Data Center* Heterogêneo — 10 Servidores @ 1 Gbps

para *data centers* homogêneos e heterogêneos, respectivamente, para todos os algoritmos estudados, com configuração `PDM`, para *data centers* com 100 servidores.

Nós constatamos que a redução no *makespan* possui, como comportamento geral, um comportamento de decaimento exponencial em função da velocidade da rede. Notavelmente, existe um limite de largura de banda após o qual não ocorre redução significativa do *makespan*. Isso ocorre tanto para *data centers* homogêneos quanto heterogêneos.

Este padrão de comportamento também se mostrou verdade para todas as outras configurações e todos os tamanhos de *data centers*.

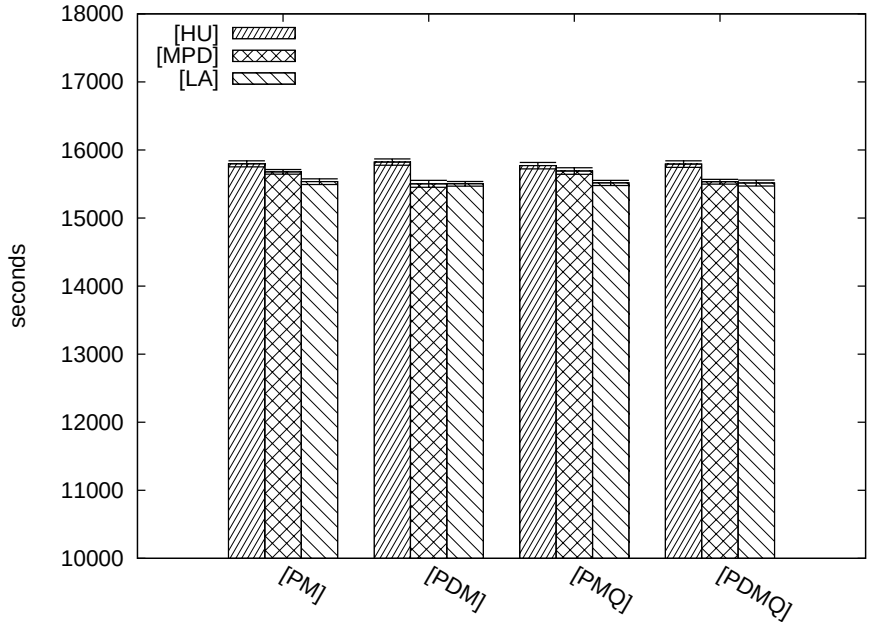

Figura 4.11: *Makespan*: *Data Center* Heterogêneo — 10 Servidores @ 100 Gbps

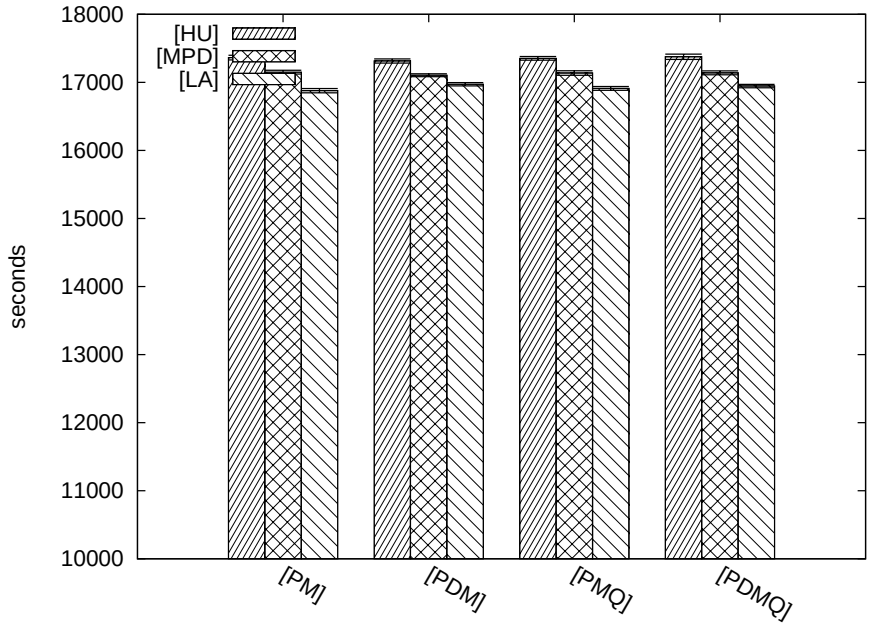

Figura 4.12: *Makespan*: *Data Center* Heterogêneo — 100 Servidores @ 1 Gbps

Considerando *data centers* homogêneos e o intervalo de confiança de 95%, nós podemos fazer as seguintes afirmações para todos os cenários estudados: em *data centers* com 10 máquinas físicas, ocorreu um *speedup* no *makespan* variando de 3% a 7% pelo aumento da largura de banda de 1 Gbps para 10 Gbps; e de 2% a 4% pelo aumento de 10 Gbps para 100 Gbps, e nenhuma melhoria significativa foi observada após este ponto. *Data centers* com 100 servidores apresentaram *speedups* variando entre 4% e 8% quando aumentando a velocidade da rede de 1 Gbps para 10 Gbps e de 1% a 3% para a melhoria de 10 Gbps para 100 Gbps, e também não houve melhoria significativa após este ponto; *data centers*

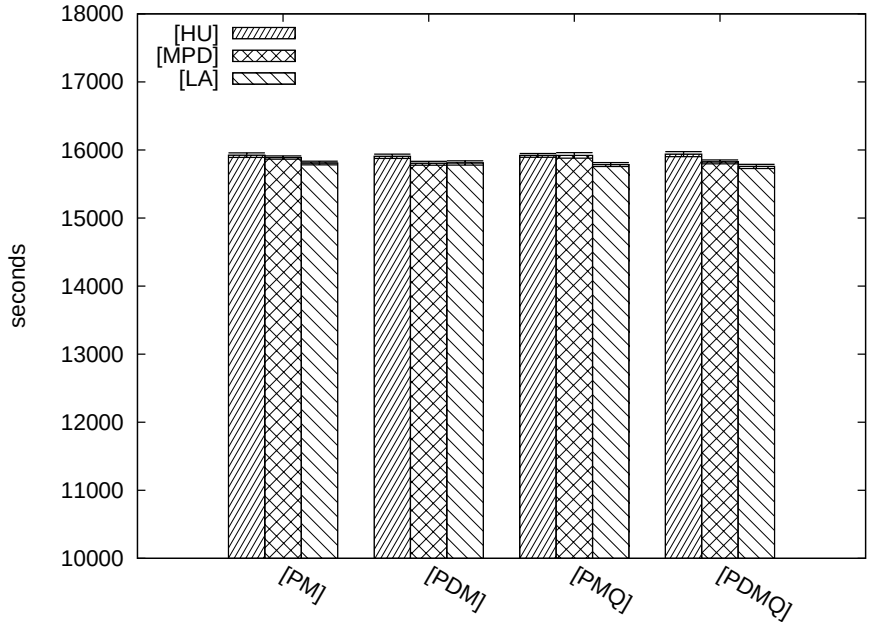

Figura 4.13: *Makespan*: *Data Center* Heterogêneo — 100 Servidores @ 100 Gbps

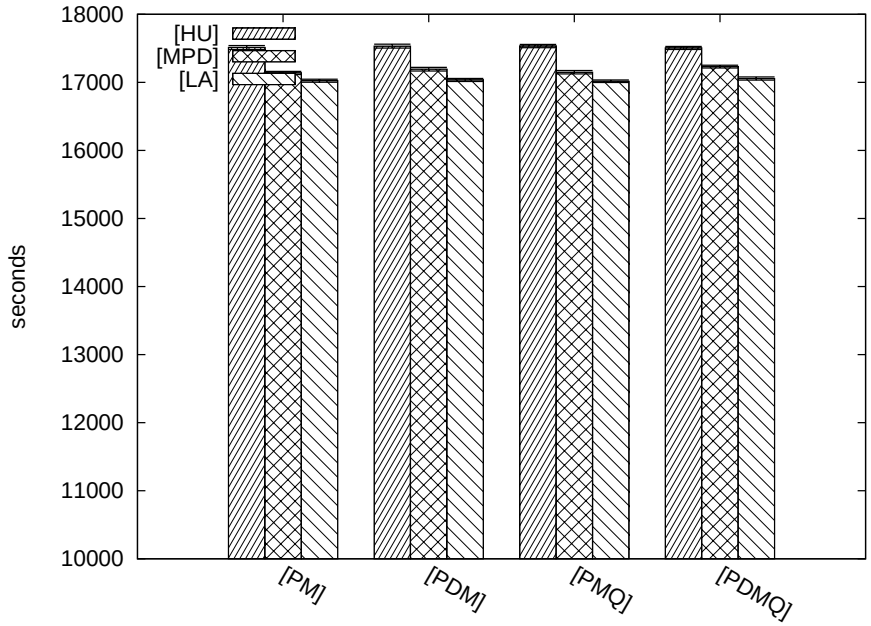

Figura 4.14: *Makespan*: *Data Center* Heterogêneo — 1.000 Servidores @ 1 Gbps

com 1.000 servidores apresentaram *speedup* de 4% a 8% com o aumento da velocidade da rede de 1 Gbps para 10 Gbps e de 2% a 3% quando a velocidade variou de 10 Gbps para 100 Gbps, também não havendo melhoria significativa após este ponto.

De modo similar, considerando *data centers* heterogêneos, também com um intervalo de confiança de 95%, nós podemos afirmar o seguinte para todos os cenários estudados: em *data centers* com 10 servidores, houve um *speedup* no *makespan* variando de 0% a 8% quando a velocidade da rede aumentou de 1 Gbps para 10 Gbps e, exceto para o algoritmo `LA`, houve uma melhoria no *makespan* de 0% a 4% quando a velocidade da rede aumentou

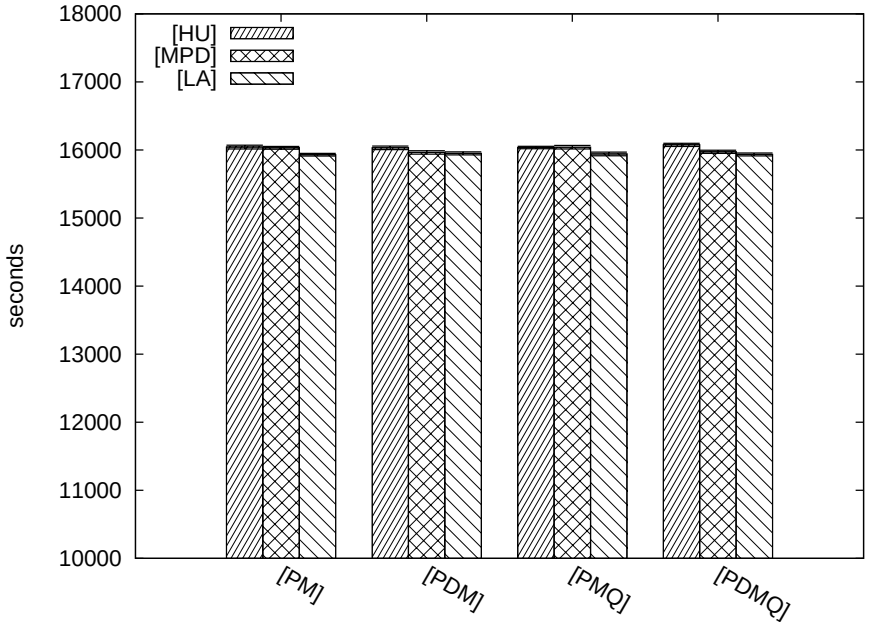

Figura 4.15: *Makespan*: *Data Center* Heterogêneo — 1.000 Servidores @ 100 Gbps

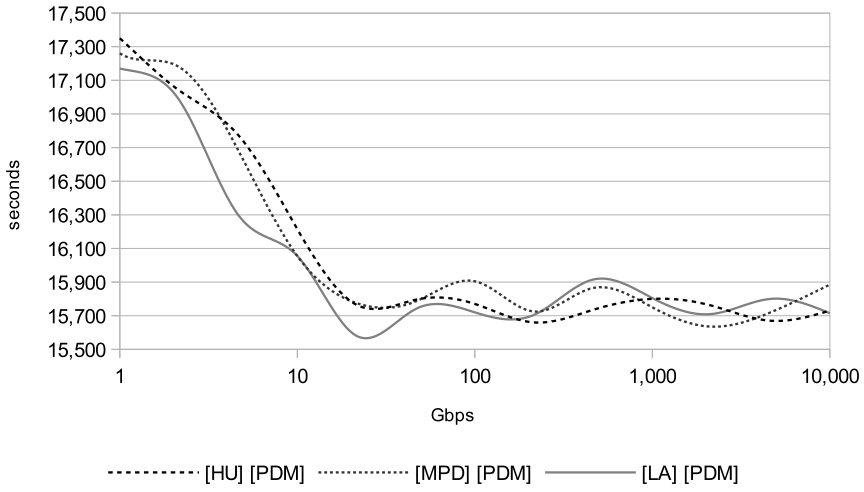

Figura 4.16: *Makespan*: *Data Center* Homogêneo — `*-PDM` — 100 Servidores

de 10 Gbps para 100 Gbps, sem melhoria significativa para todos os demais casos. No caso de *data centers* com 100 servidores, considerando o aumento da velocidade da rede de 1 Gbps para 10 Gbps, nós observamos uma melhoria variando de 5% a 8%. Exceto para o `LA`, o aumento da velocidade de 10 Gbps para 100 Gbps trouxe uma melhoria de 2% a 4%, e não houve melhoria significativa para todos os demais cenários. Para *data centers* com 1.000 servidores, nós observamos uma melhoria de 4% a 7% pelo aumento da velocidade de 1 Gbps para 10 Gbps, e de 0% a 4% pelo aumento de 10 Gbps para 100 Gbps, sem observar melhorias significativas após este ponto.

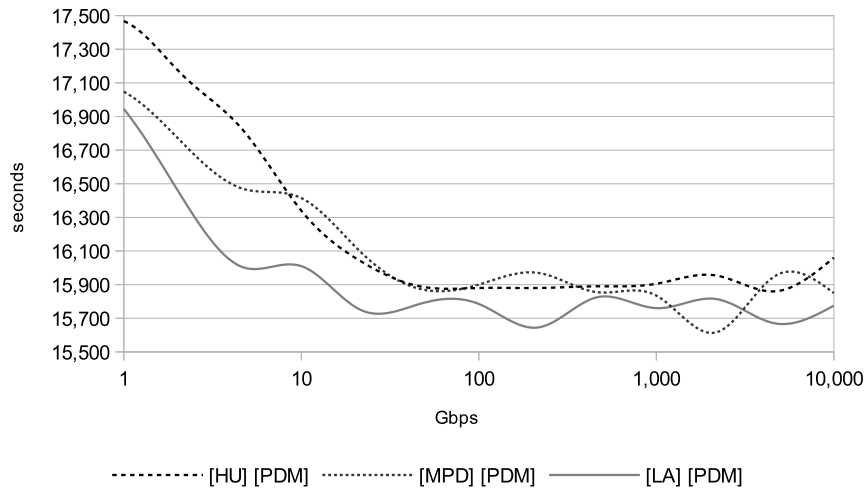

Figura 4.17: *Makespan*: *Data Center* Heterogêneo — `*-PDM` — 100 Servidores

4.2.4 Resultados: Tempos de Execução de *Cloudlets*

Nós apresentamos os tempos de execução das *cloudlets* para os algoritmos de escalonamento de máquinas virtuais `HU`, `MPD` e `LA` sob as configurações `PM`, `PDM`, `PMQ` e `PDMQ` nas Figuras 4.18–4.23 para duas diferentes velocidades de redes, 1 Gbps e 100 Gbps, relembrando que existem 3 prioridades de *cloudlets*: alta, normal e baixa.

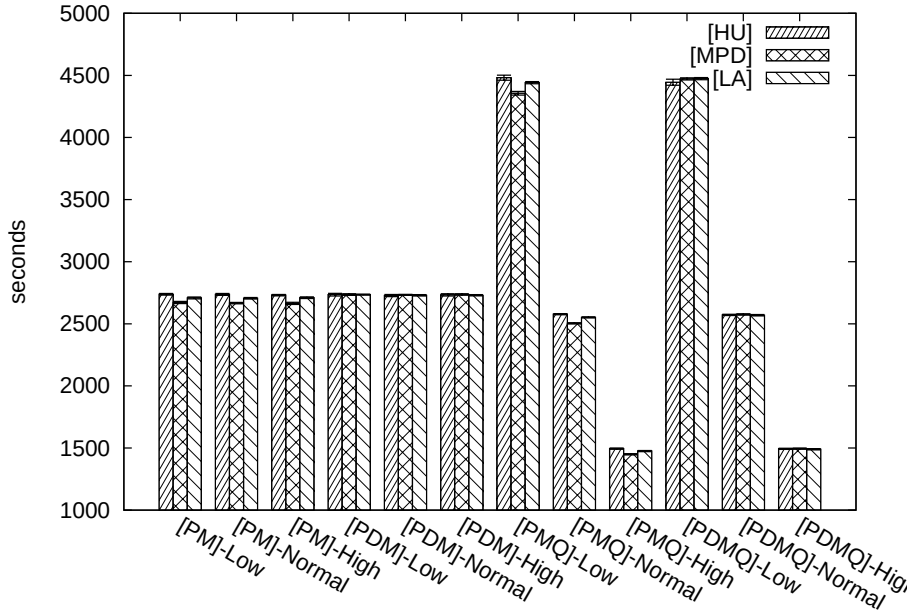

Figura 4.18: Tempos de Execução de *Cloudlets*: *Data Center* Homogêneo — 10 Servidores @ 1 Gbps

Nós observamos que no caso das cargas de trabalho, houve uma redução nos tempos com as velocidades de rede mais rápidas, como ocorreu no caso do *makespan*. Uma possível interpretação para isso é por que a migração reduz o desempenho de máquinas virtuais que processam cargas de trabalho, e em consequência estas passam a apresentar tempos de processamento aumentados.

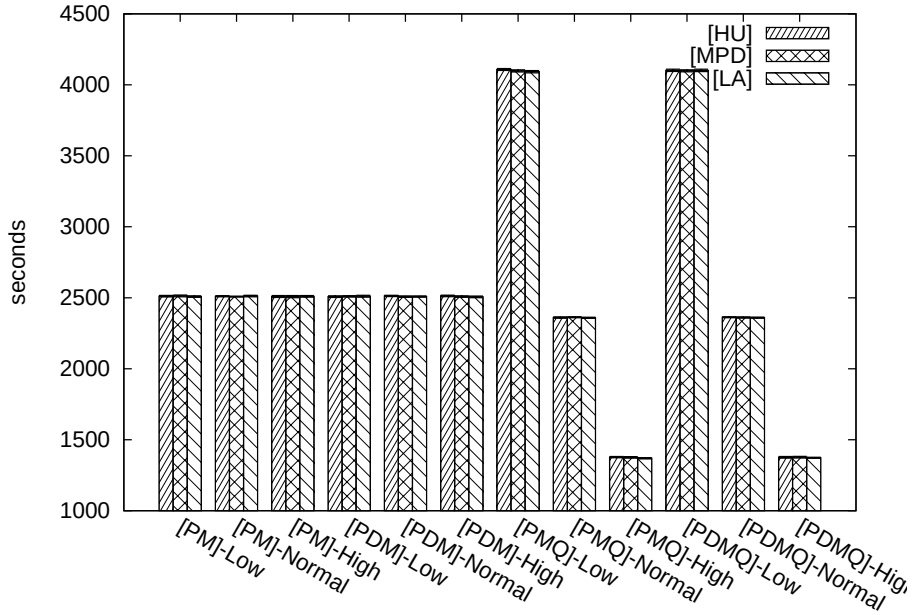

Figura 4.19: Tempos de Execução de *Cloudlets*: *Data Center* Homogêneo — 10 Servidores @ 100 Gbps

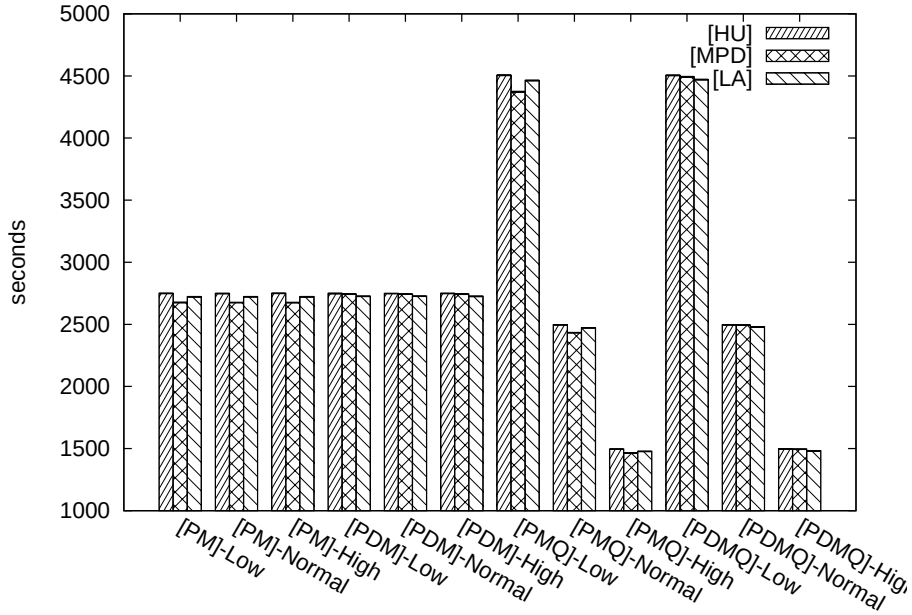

Figura 4.20: Tempos de Execução de *Cloudlets*: *Data Center* Homogêneo — 100 Servidores @ 1 Gbps

Nós observamos que o impacto das migrações no processamento das cargas é grande o bastante para também se manifestar no *makespan* (global). Isso reforça a ideia que o *makespan* pode ser melhorado em decorrência de migrações mais rápidas, que possibilitam menor tempo do processo da migração para máquinas virtuais — com sua respectiva degradação de desempenho — reduzindo os tempos de processamento destas cargas, que consolidados permitem a redução do *makespan*.

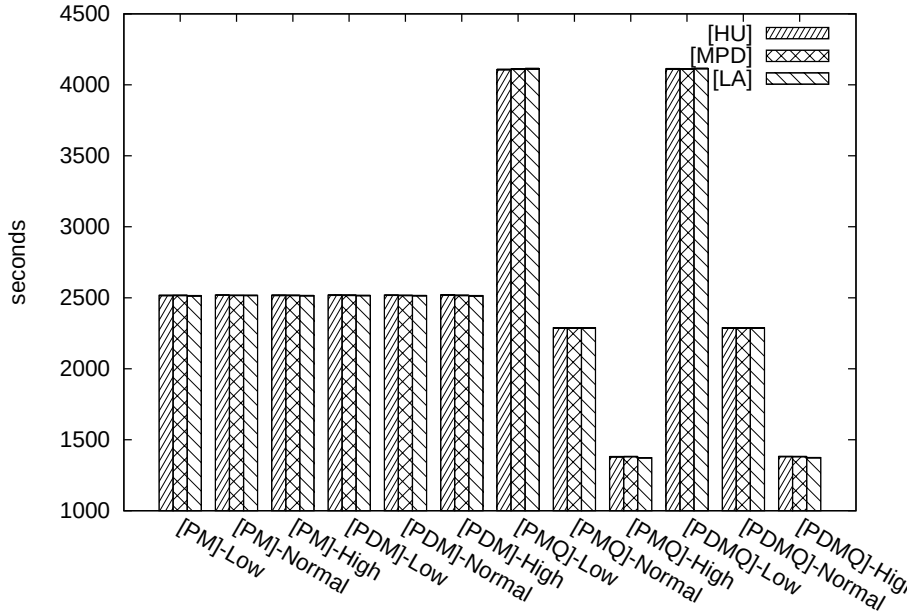

Figura 4.21: Tempos de Execução de *Cloudlets*: *Data Center* Homogêneo — 100 Servidores @ 100 Gbps

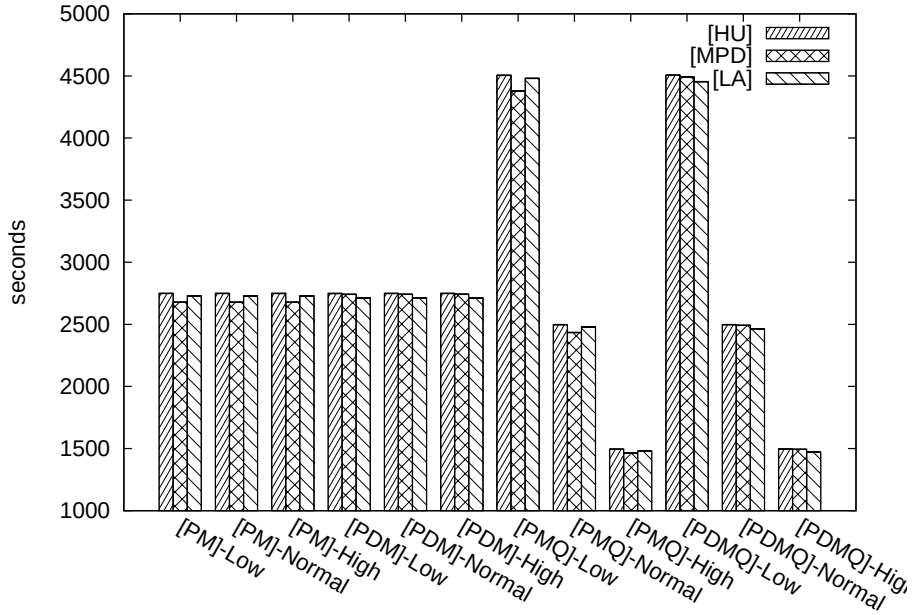

Figura 4.22: Tempos de Execução de *Cloudlets*: *Data Center* Homogêneo — 1.000 Servidores @ 1 Gbps

Nós apresentamos nas Figuras 4.24–4.29 os tempos de execução dos algoritmos de escalonamento de máquinas virtuais `HU`, `MPD` e `LA` sob as configurações `PM`, `PDM`, `PMQ` e `PDMQ` para *data centers* heterogêneos com velocidades de redes de 1 Gbps e 100 Gbps. Como podemos observar, existe uma notável redução nos tempos de execução das cargas com o aumento da largura de banda de 1 Gbps para 100 Gbps.

A seguir, nós consideramos o impacto no tempo de execução em função da velocidade da rede. Na Figura 4.30, nós plotamos os tempos das cargas de trabalho dependendo

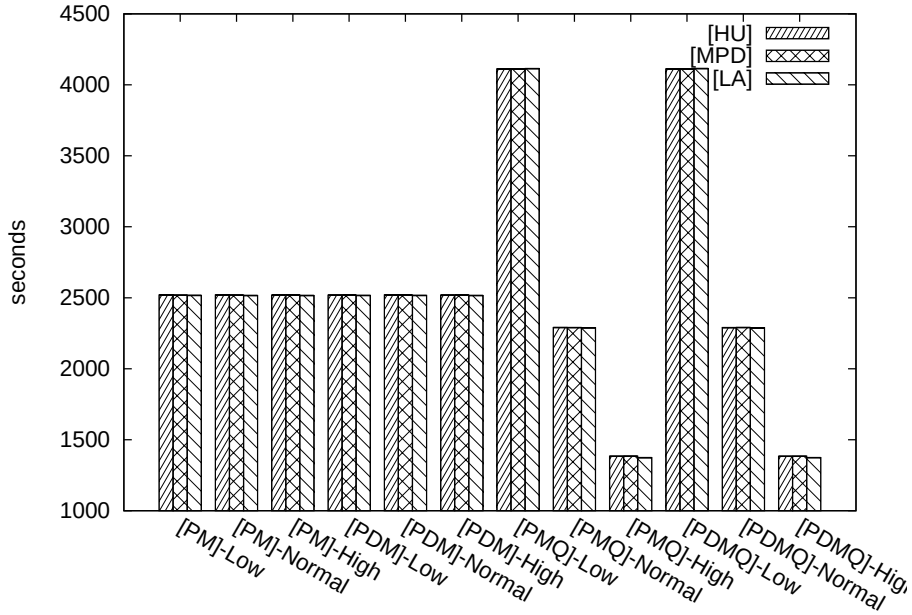

Figura 4.23: Tempos de Execução de *Cloudlets*: *Data Center* Homogêneo — 1.000 Servidores @ 100 Gbps

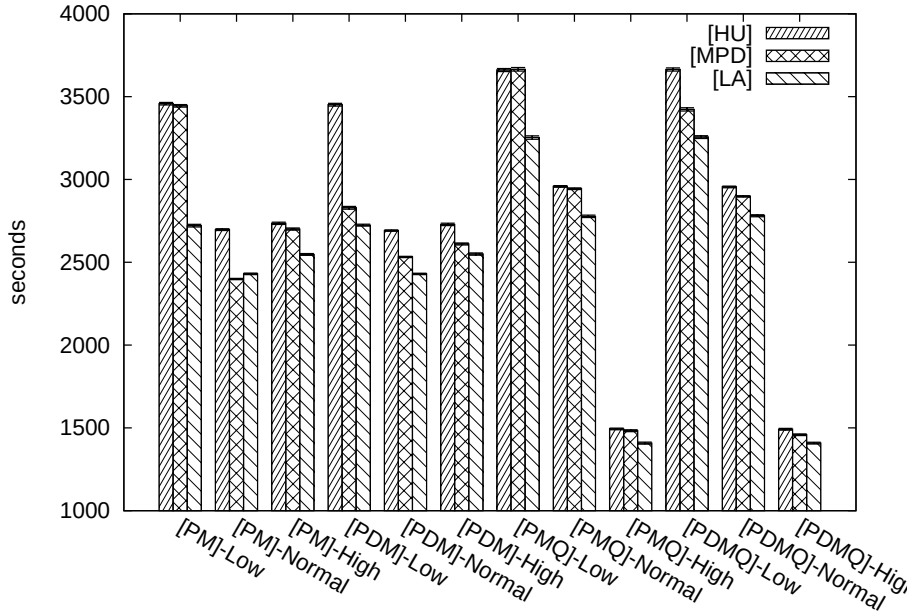

Figura 4.24: Tempos de Execução de *Cloudlets*: *Data Center* Heterogêneo — 10 PMs @ 1 Gbps

da velocidade das redes para *data centers* homogêneos, e para *data centers* heterogêneos na Figura 4.31. Em ambos os casos, nós usamos todos os algoritmos estudados com a configuração `PM` para *data centers* com 100 servidores.

Semelhante ao padrão de comportamento do *makespan*, nós verificamos que o decaimento exponencial também ocorre para os tempos de execução das cargas de trabalho. Em outras palavras, o aumento da velocidade das redes também reduziu os tempos de execução das cargas de trabalho, mas existe um limite superior de velocidade após o qual

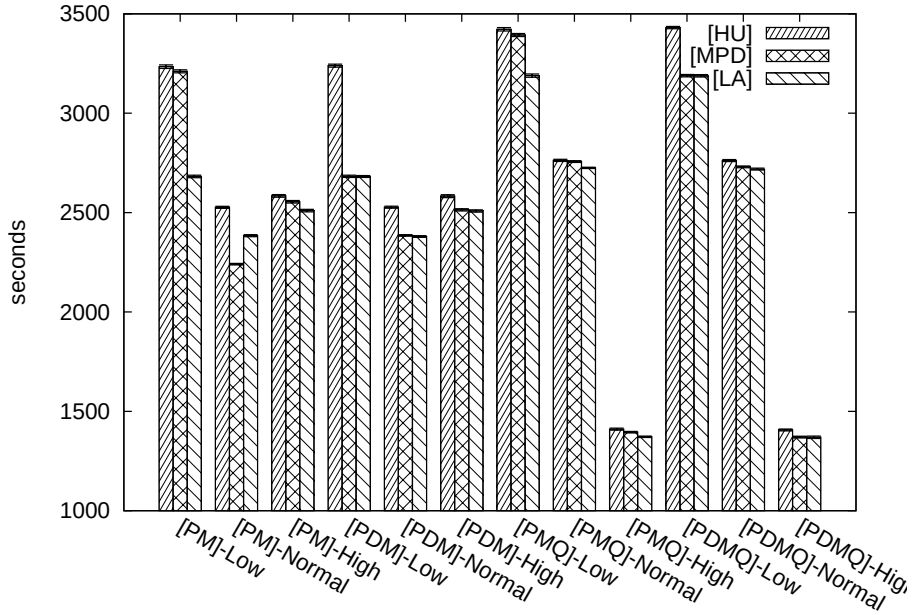

Figura 4.25: Tempos de Execução de *Cloudlets*: *Data Center* Heterogêneo — 10 PMs @ 100 Gbps

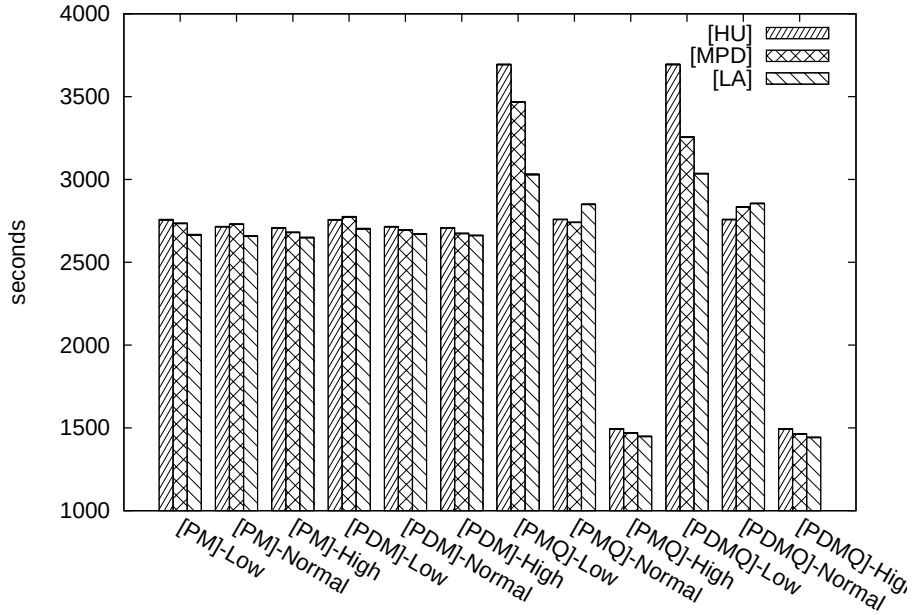

Figura 4.26: Tempos de Execução de *Cloudlets*: *Data Center* Heterogêneo — 100 PMs @ 1 Gbps

nós não observamos maior redução nos tempos de execução. Este padrão é o mesmo para *data centers* homogêneos e heterogêneos, para todos os algoritmos e para todas as configurações.

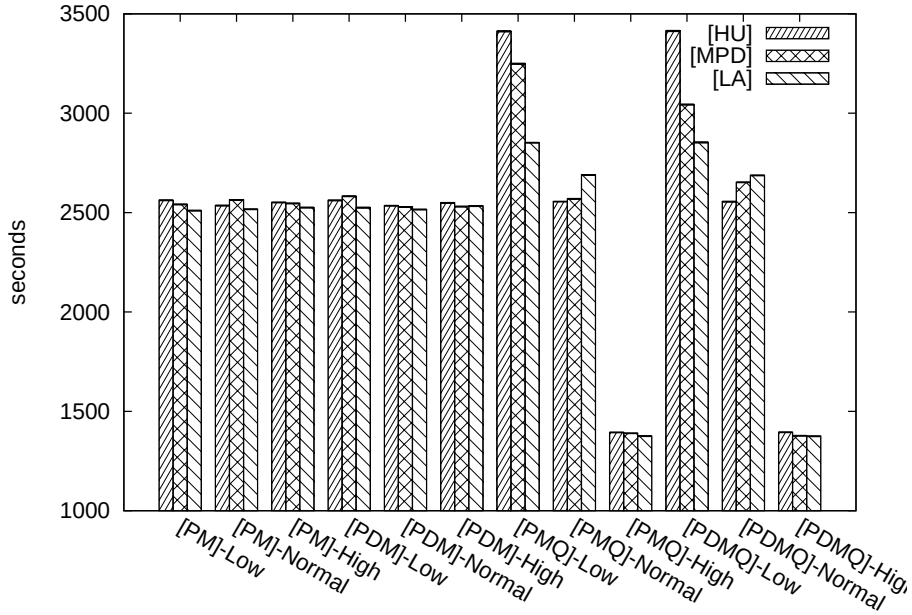

Figura 4.27: Tempos de Execução de *Cloudlets*: *Data Center* Heterogêneo — 100 PMs @ 100 Gbps

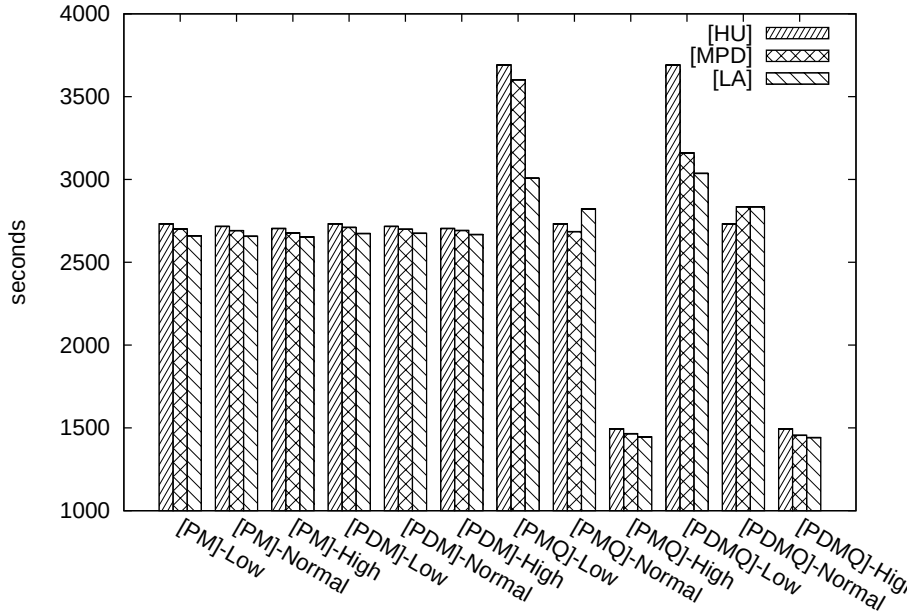

Figura 4.28: Tempos de Execução de *Cloudlets*: *Data Center* Heterogêneo — 1.000 PMs @ 1 Gbps

4.2.5 Resultados: Número de Migrações

Por fim, nós consideramos o impacto do aumento da velocidade das redes no *número de migrações*. Nós apresentamos o número de migrações de máquinas virtuais para os algoritmos de escalonamento `HU`, `MPD` e `LA` sob as configurações `PM`, `PDM`, `PMQ` e `PDMQ` nas Figuras 4.32–4.37 para velocidades de redes de 1 Gbps e 100 Gbps. Para um pequeno número de servidores, o número de migrações decresce à medida que a velocidade da rede

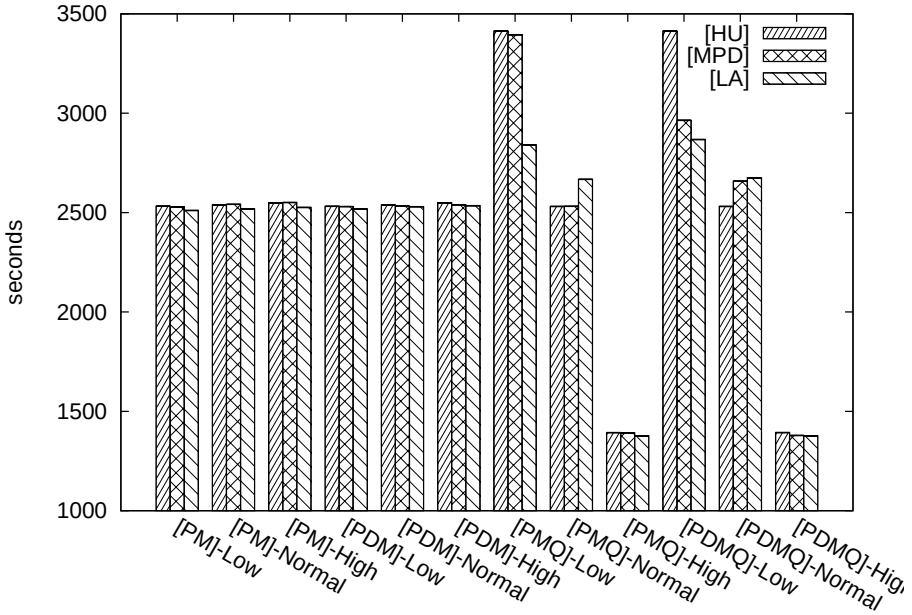

Figura 4.29: Tempos de Execução de *Cloudlets*: *Data Center* Heterogêneo — 1.000 PMs @ 100 Gbps

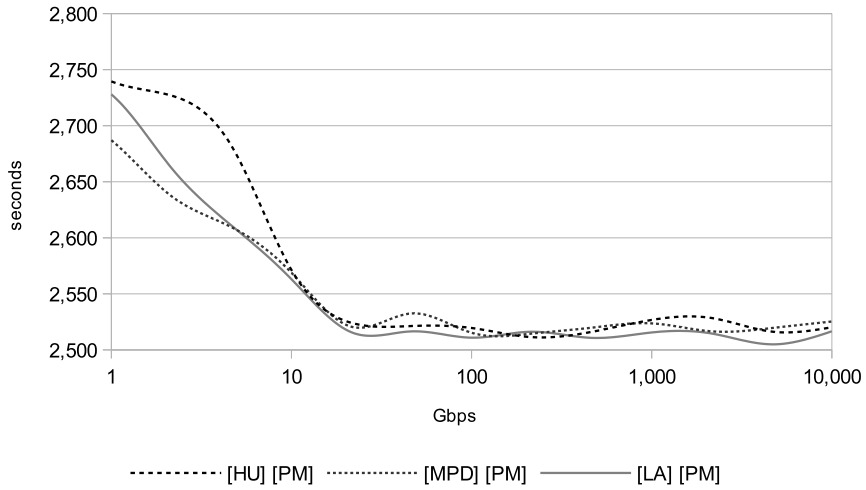

Figura 4.30: Tempos de Execução de *Cloudlets*: Homo. DC — `*-PM` — 100 PMs

aumenta. No entanto, para um grande número de servidores, esta alteração depende do algoritmo de escalonamento em questão.

Para *data centers* heterogêneos, o número de migrações de máquinas virtuais para os algoritmos de escalonamento `HU`, `MPD` e `LA` sob as configurações `PM`, `PDM`, `PMQ` e `PDMQ` são apresentados nas Figuras 4.38–4.43.

Nós observamos que um intrigante e inesperado comportamento ocorreu com respeito ao número de migração de máquinas virtuais em função da velocidade da rede. Diferentemente do *makespan* e dos tempos de execução das *cloudlets*, nós não observamos um decaimento exponencial aqui. Quando nós analisamos o impacto da rede no número de migrações, nós observamos que aumentando a velocidade das redes existe um aumento do número de migrações até um certo limite, após o qual existe um declínio acentuado

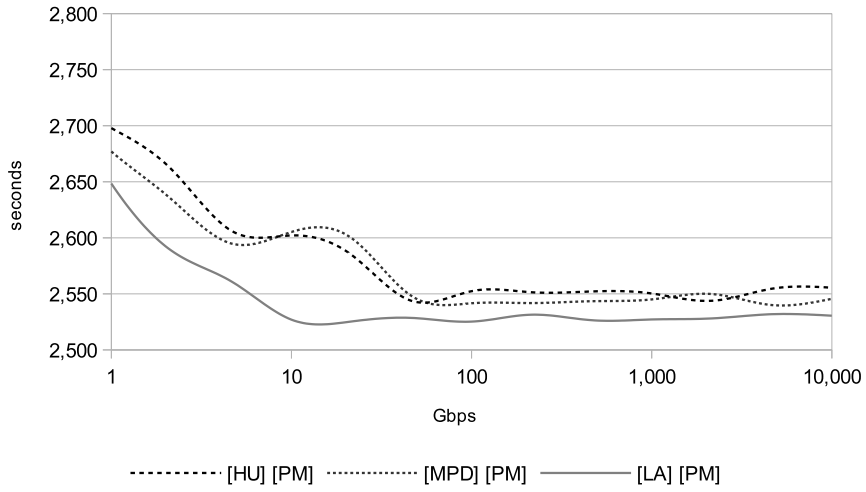

Figura 4.31: Tempos de Execução de *Cloudlets*: Het. DC — `*-PM` — 100 PMs

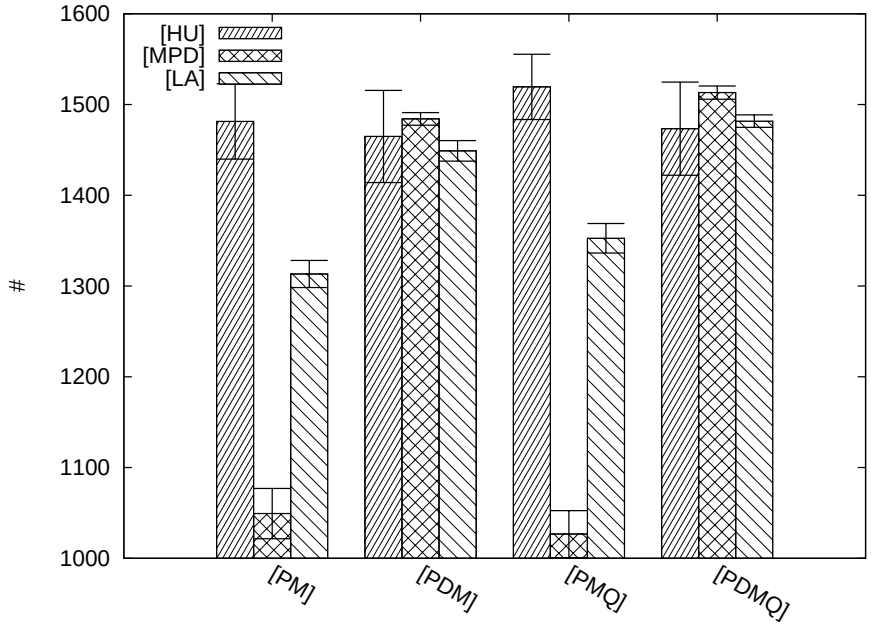

Figura 4.32: Número de Migrações: *Data Center* Homogêneo — 10 Servidores @ 1 Gbps

e uma estabilização. Nós exemplificamos pontos amostrados do número de migrações de acordo com a largura de banda para *data centers* homogêneos na Figura 4.44 e para *data centers* Heterogêneos na Figura 4.45. Em ambos os casos, nós usamos todos os algoritmos de escalonamento de máquinas virtuais com a configuração `PM` e 100 servidores. Nós não podemos concluir que um número maior ou menor de migrações é necessariamente bom ou ruim, uma vez que ele não impacta necessariamente no consumo de energia, *makespan* ou tempo de execução de cargas individuais. Este comportamento é semelhante para *data centers* homogêneos e heterogêneos e todos os algoritmos estudados com todas as configurações. Para todos os algoritmos com todas as configurações apresentadas, a taxa de pico de número de migrações ocorreu entre 1 Gbps e 100 Gbps, indicando que nos cenários

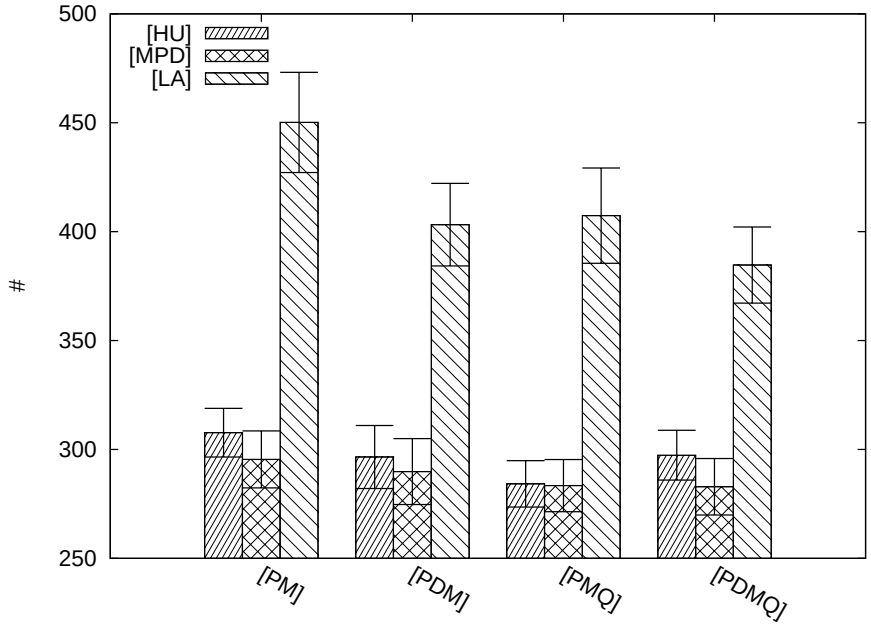

Figura 4.33: Número de Migrações: *Data Center* Homogêneo — 10 Servidores @ 100 Gbps

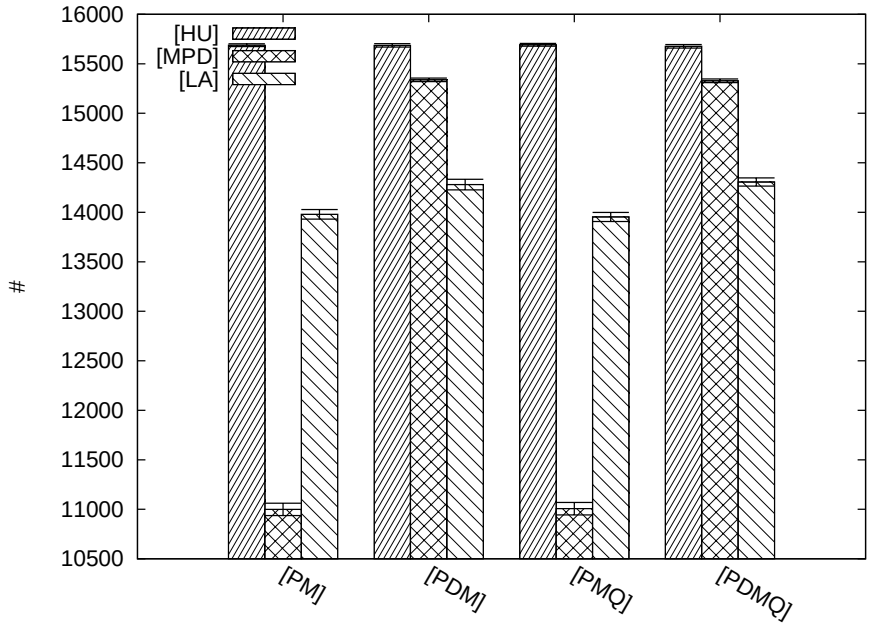

Figura 4.34: Número de Migrações: *Data Center* Homogêneo — 100 Servidores @ 1 Gbps

estudados taxas acima de 100 Gbps não aumentam a taxa de migrações, em contraste, mantêm este nível estável.

Nossa reflexão inicial foi que o motivo para este comportamento não usual no número de migrações foi devido à dependência do algoritmo de escalonamento. Para entender isso, considere, por exemplo, três servidores, $h1$, $h2$ e $h3$, com eficiências em energia alta, média e baixa, respectivamente — i.e., $h1$ é a máquina mais eficiente em energia — e cada uma das máquinas virtuais $vm1$, $vm2$ e $vm3$ estão, respectivamente, alocadas em $h1$, $h2$ e $h3$. Considere também que, quanto à finalização das cargas processadas pelas

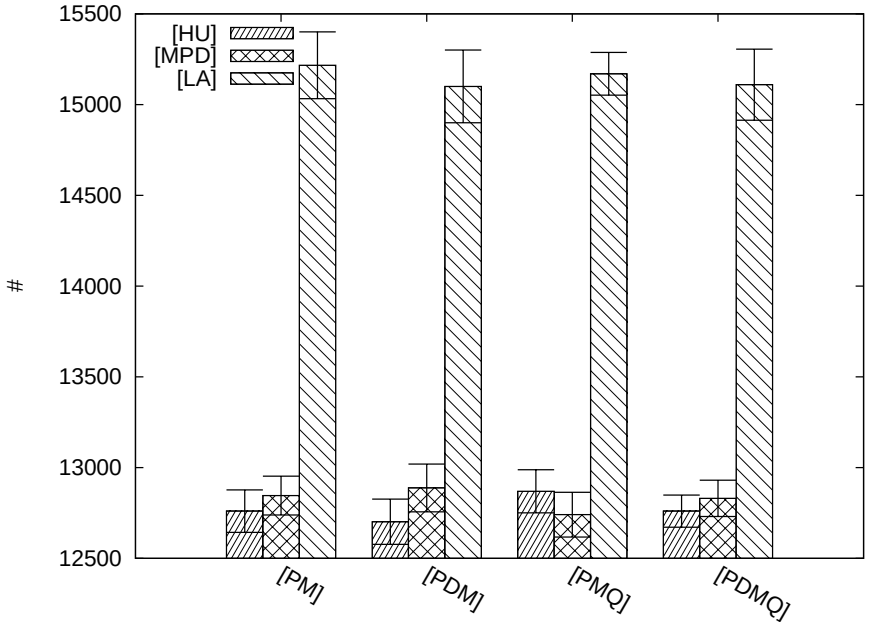

Figura 4.35: Número de Migrações: *Data Center* Homogêneo — 100 Servidores @ 100 Gbps

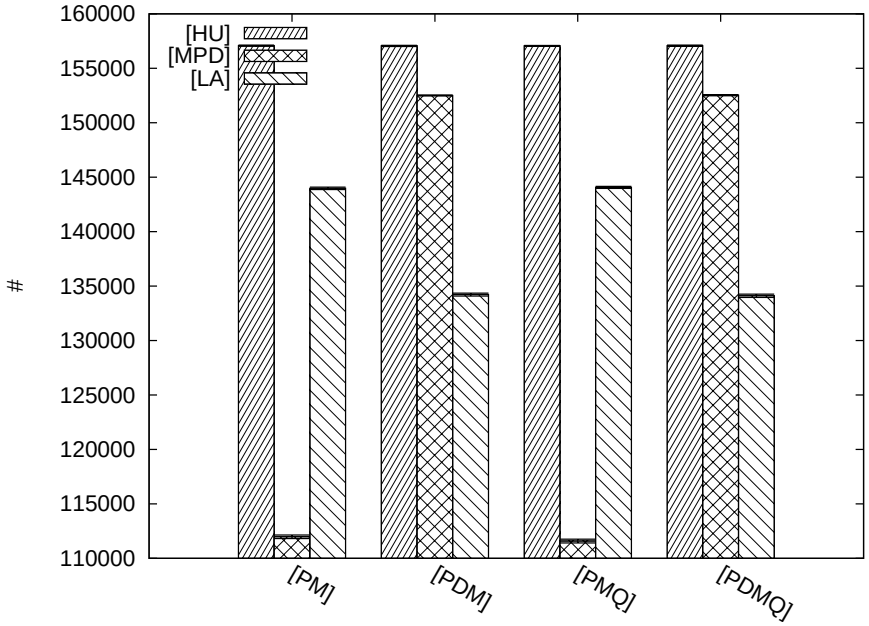

Figura 4.36: Número de Migrações: *Data Center* Homogêneo — 1.000 Servidores @ 1 Gbps

máquinas virtuais, $vm1$ termina primeiro, seguida pela $vm2$ e finalmente $vm3$. Considere ser possível realizar migrações, e que as máquinas virtuais durante as migrações podem processar (apesar de experimentarem uma degradação de desempenho).

Considere primeiramente um tempo de migração alto. Suponha que $vm1$ finaliza seu processamento, então a $vm2$ será migrada de $h2$ para $h1$, e na sequência $vm3$ migra de $h3$ para $h2$. Se durante a migração de $vm3$ para $h2$, $vm2$ finalizar seu processamento, então $vm3$ pode não ser imediatamente migrada para $h1$ (o servidor mais eficiente em energia),

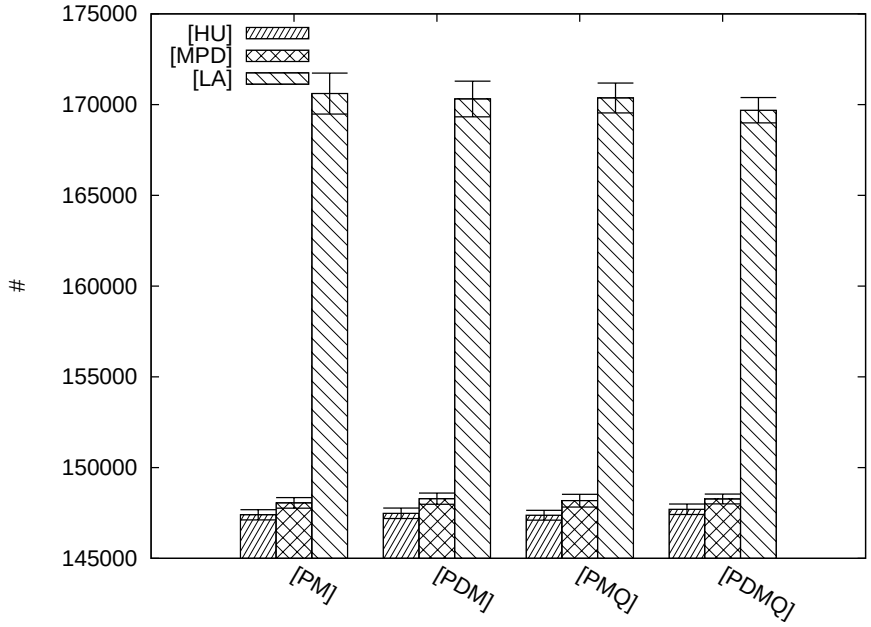

Figura 4.37: Número de Migrações: *Data Center* Homogêneo — 1.000 Servidores @ 100 Gbps

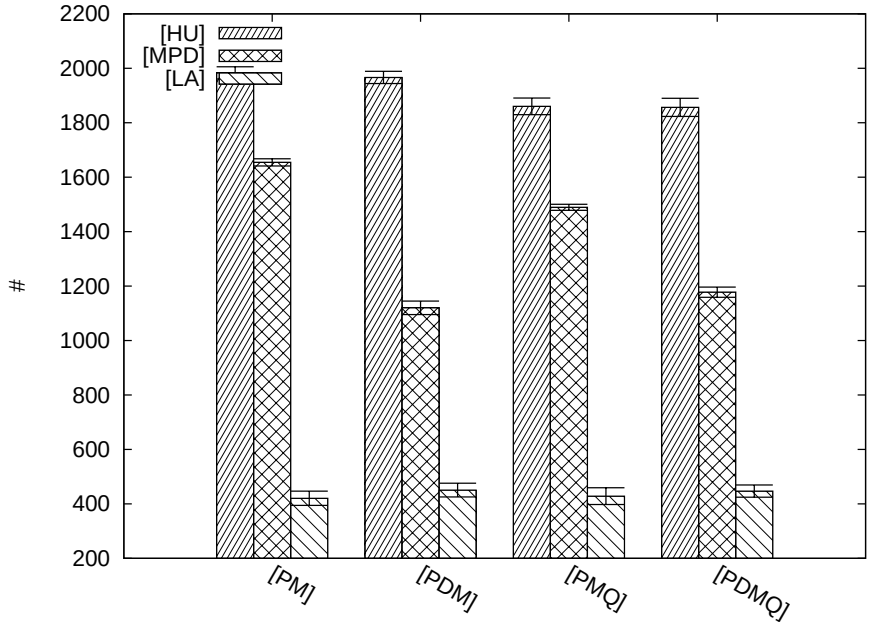

Figura 4.38: Número de Migrações: *Data Center* Heterogêneo — 10 Servidores @ 1 Gbps

pois ainda estará em migração para $h2$. Apenas após a migração de $vm3$ para $h2$ estar concluída é que $vm3$ poderá ser migrada para $h1$, fazendo com que este tempo restante de migração para $h2$ resulte em uma ineficiência sistêmica em energia, *makespan* e tempo de execução de cargas, pois é necessário manter $h2$ ligada durante a migração enquanto $vm3$ deveria estar sendo migrada para $h1$. Esta migração também pode levar à degradação de desempenho no processamento de $vm3$. E mais do que isso, este longo tempo consumido pelas migrações implica em um menor número de migrações.

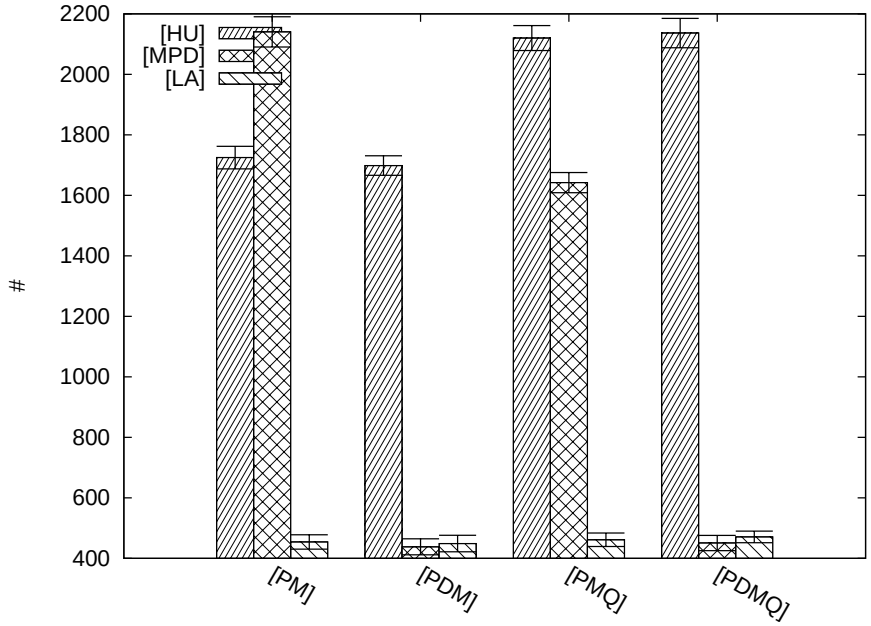

Figura 4.39: Número de Migrações: *Data Center* Heterogêneo — 10 Servidores @ 100 Gbps

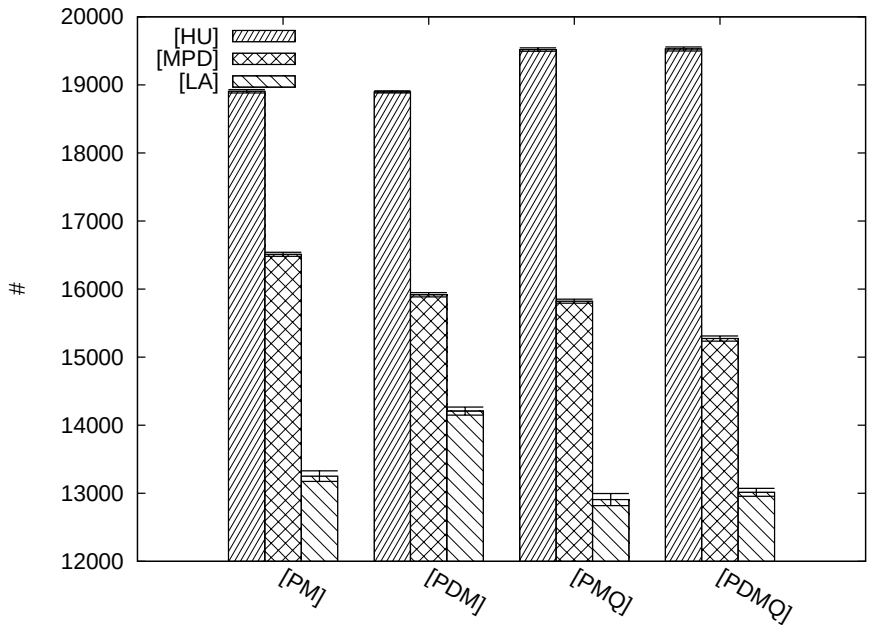

Figura 4.40: Número de Migrações: *Data Center* Heterogêneo — 100 Servidores @ 1 Gbps

Considere, a seguir, para o mesmo cenário, que os tempos de migrações são baixos — e.g. que tenhamos redes mais rápidas nos *data centers*. Suponha que após a conclusão de $vm1$, $vm2$ será migrada de $h2$ para $h1$ e seguida por $vm3$ de $h3$ para $h2$. O tempo de migração de $vm3$ para $h1$ neste caso será substancialmente reduzido, pois as redes de alta velocidade farão com que a migração de $vm3$ para $h2$ ocorra em uma taxa acelerada. Em outras palavras, redes de alta velocidade favorecem migrações mais rápidas, que implicam em disponibilidade mais rápida de servidores para máquinas virtuais.

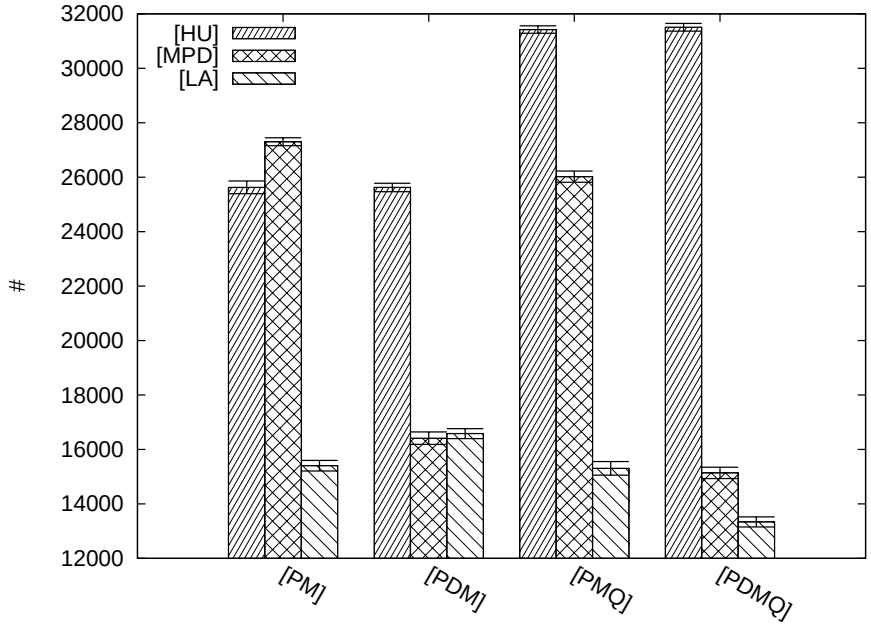

Figura 4.41: Número de Migrações: *Data Center* Heterogêneo — 100 Servidores @ 100 Gbps

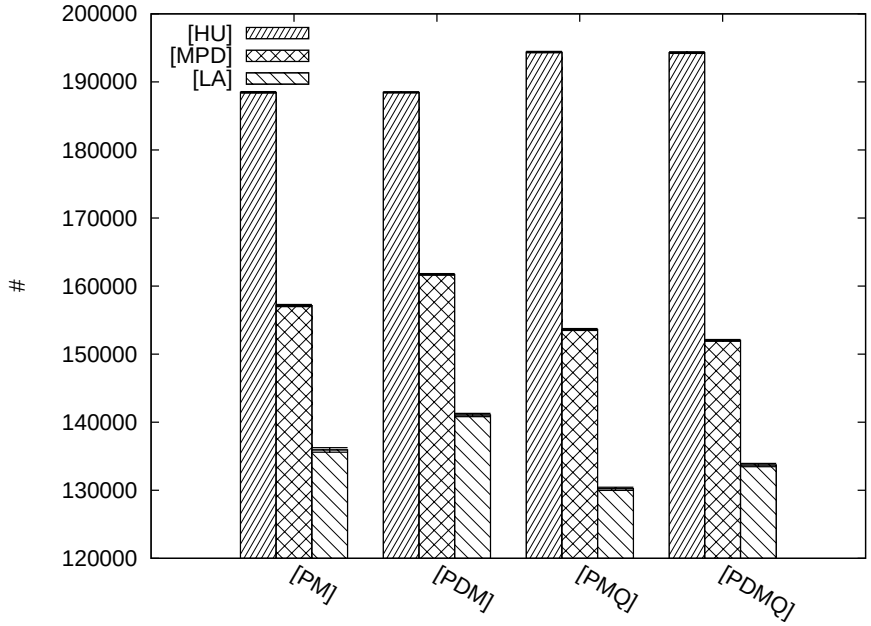

Figura 4.42: Número de Migrações: *Data Center* Heterogêneo — 1.000 Servidores @ 1 Gbps

Portanto, redes de alta velocidade aumentam a disponibilidade de servidores para máquinas virtuais e também reduzem os tempos de migração, o que permite que as máquinas virtuais permaneçam por mais tempo alocadas em um servidor — e não migrando entre dois servidores. Isso possibilita um melhor manejo de cargas para as máquinas virtuais. Agora, ainda resta a seguinte questão para ser respondida: para os cenários avaliados, por que o número de migrações é reduzido drasticamente e estabiliza após um determinado ponto?

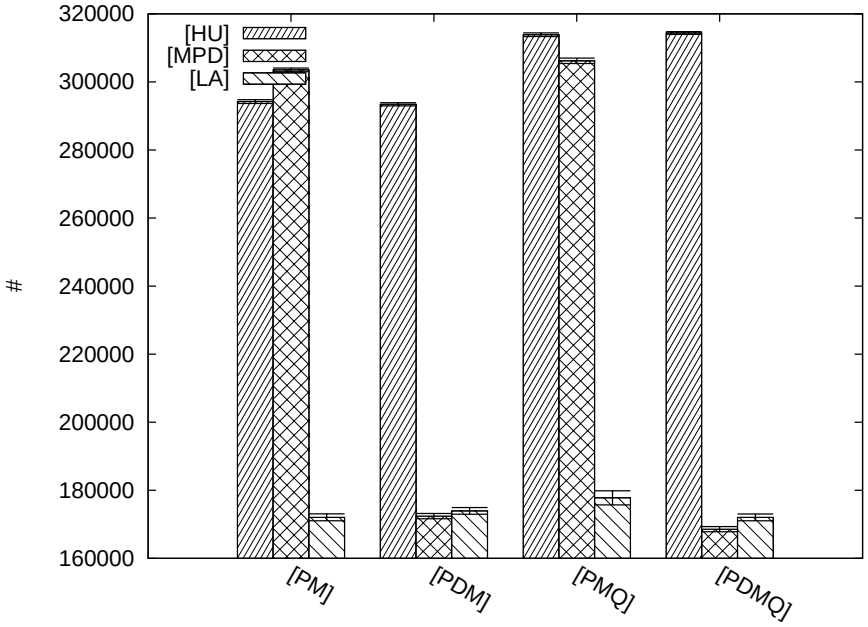

Figura 4.43: Número de Migrações: *Data Center* Heterogêneo — 1.000 Servidores @ 100 Gbps

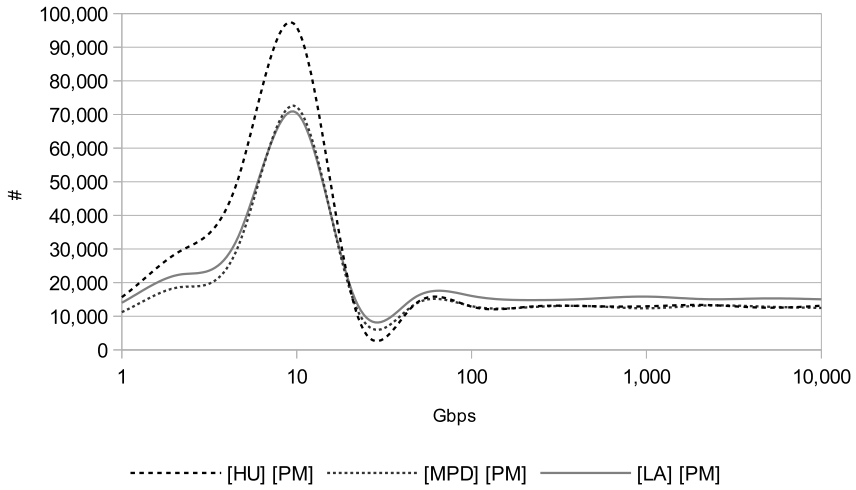

Figura 4.44: Número de Migrações: *Data Center* Homogêneo — `*-PM` — 100 Servidores

Duas hipóteses parecem ser plausíveis para esta questão: (i) devido à maior disponibilidade de servidores, o algoritmo de escalonamento pode realizar alocações otimizadas; ou (ii) o tempo de processamento das máquinas virtuais, por não sofrer o impacto de desempenho durante o tempo de migração, melhora tanto o processamento das cargas que permite que as máquinas virtuais sejam terminadas mais cedo, fazendo com que um maior número de migrações de máquinas virtuais não precisem ocorrer.

Para responder esta questão, nós implementamos um algoritmo de migração continuada (`RR`) e consideramos uma degradação de desempenho de processamento de 10% durante a migração das máquinas virtuais (valor padrão adotado no *CloudSim*). Este algoritmo objetiva selecionar servidores para máquinas virtuais em alternância circular e como resultado mantém, salvo condições muito específicas (quando o número de máqui-

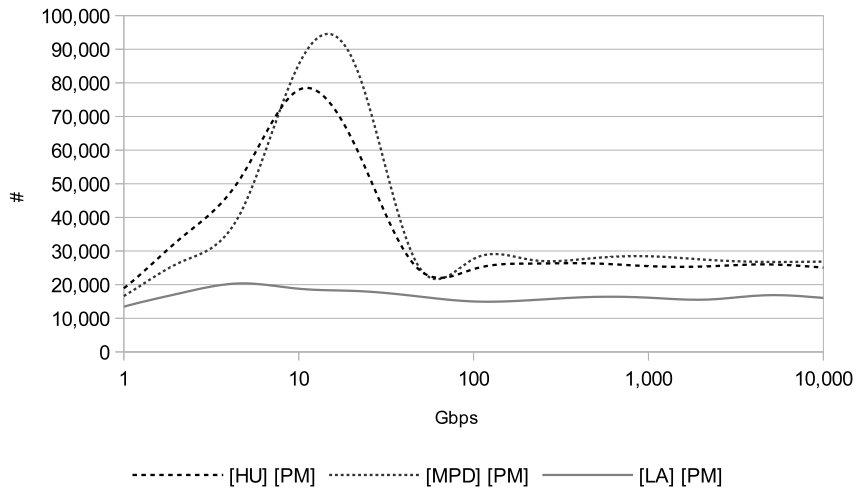

Figura 4.45: Número de Migrações: *Data Center* Heterogêneo — `*-PM` — 100 Servidores

nas virtuais seja múltiplo do número de servidores), as máquinas virtuais migrando por praticamente todo o seu tempo de vida. Se o número de migrações continua a subir, isso indica que a hipótese (i) é verdadeira, caso contrário, é uma indicação que a hipótese (ii) é a correta. Apresentamos na Figura 4.46 o comportamento do número de migrações em função da velocidade da rede plotado para o algoritmo `RR-PM` para um *data center* homogêneo. Como podemos observar, existe um número de migrações de máquinas virtuais crescente, sem decaimento, e com estabilização ao final. Isso não indica um decaimento no número de máquinas virtuais processando cargas, sugerindo que a hipótese (ii) não seja verdadeira. Isto é, nós temos uma indicação que a alteração no número de migrações de máquinas virtuais é muito mais dependente do algoritmo usado no processo de escalonamento de máquinas virtuais do que na disponibilidade de servidores para processamento de máquinas virtuais resultante de uma migração mais rápida destas máquinas virtuais.

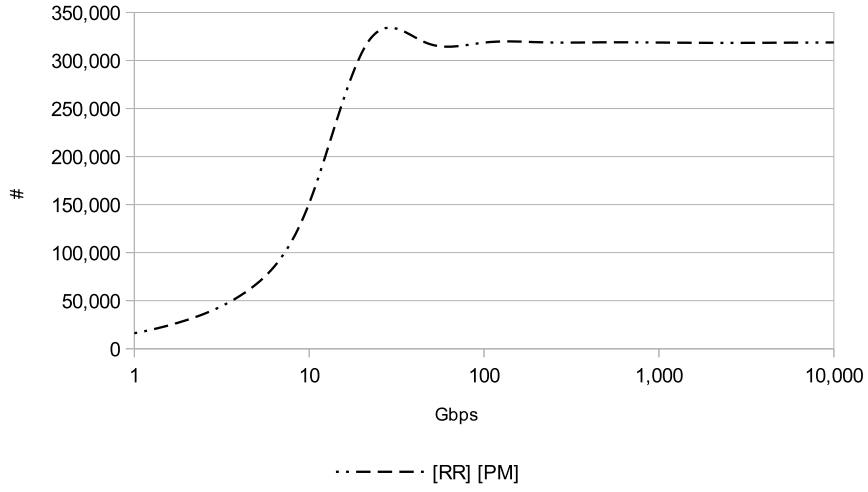

Figura 4.46: Número de Migrações: *Data Center* Homogêneo — `RR-PM` — 100 Servidores

4.3 Comentários Finais

Neste capítulo, constatamos que o aumento da velocidade da rede influencia o consumo de energia no processo de escalonamento das máquinas virtuais em *data centers* para os cenários estudados. Observamos que todos os algoritmos de escalonamento foram impactados positivamente, pois, considerando constante o consumo de energia dos *switches*, os algoritmos tendem a reduzir o consumo de energia com uma velocidade de rede maior. Também demonstramos como calcular se um aumento de largura de banda no *data center* vale a pena em termos de eficiência energética. Os resultados obtidos por simulação mostram que os benefícios dependem do algoritmo usado. Considerando larguras de banda de até 10 Tbps, o algoritmo mais eficiente em energia em *data centers* (i) grandes e homogêneos, (ii) médios e heterogêneos e (iii) grandes e heterogêneos foi o `LA`.

Também desenvolvemos um modelo empírico usando uma função de decaimento exponencial que se ajustou bem aos cenários estudados. Este modelo nos permite observar que, para os quatro algoritmos de escalonamento, existe um limite superior, para a maioria dos casos de 100 Gbps, a partir do qual a largura de banda aumentada não afeta o consumo de energia.

Nós também apresentamos e analisamos o impacto da velocidade da rede do *data center* em *data centers* que operam na nuvem em relação ao número de migrações, *makespan* e tempos de execução das cargas de trabalho individuais. Para este estudo, aprimoramos o *CloudSim* modificando os algoritmos de escalonamento de máquinas virtuais `HU` e `MPD`, de modo a trabalharem com um algoritmo de escalonamento de cargas de trabalho com QoS baseado em prioridade de cargas [76].

Descobrimos que, para todos os cenários avaliados, o aumento da largura de banda reduz os tempos de execução do *makespan* e das cargas de trabalho com um comportamento de decaimento exponencial à medida que a velocidade da rede aumenta. Nós também observamos que o mecanismo de migração *live* geralmente envolve uma degradação do desempenho de processamento, mas essa degradação afeta o *makespan* menos do que o algoritmo de escalonamento usado.

Também descobrimos que o número de migrações varia de acordo com o algoritmo de escalonamento empregado, mas o comportamento geral é um aumento no número de migrações até atingir um pico, geralmente entre 1 Gbps e 100 Gbps, seguido de uma queda acentuada até este número atinge um ponto de estabilidade. No entanto, percebemos que o número de migrações tem pouco impacto nos fatores importantes, como o consumo de energia ou os tempos de processamento.

Capítulo 5

Bandwidth-Aware Lago Allocator

Nossos estudos sobre os impactos do contínuo aumento das taxas de transmissão do padrão *Ethernet* concluíram que esta evolução pode trazer eficiência em energia, pois, entre outras observações, ao efetuar migrações mais rápidas de máquinas virtuais é possível desligar mais servidores ociosos em menos tempo.

Motivados pelo fato que a migração *live*, apesar de usualmente envolver na degradação do desempenho de processamento, que esta degradação afeta menos o *makespan* do que o algoritmo utilizado, sentimos segurança em dar um próximo passo: desenvolver um método de escalonamento ciente de energia e de largura de banda para prover eficiência em energia.

Alguns fatores norteiam o algoritmo desenvolvido. Primeiramente, a rápida evolução das taxas de transmissão do protocolo *Ethernet* implica na dificuldade de implantação de redes no mesmo ritmo, isto é, favorece a existência de grupos de servidores, mais antigos, com taxas de transmissão menores, coexistindo com grupos de servidores, mais novos, com taxas de transmissão maiores. Portanto, em um senso realístico, é muito provável, inclusive em grande corporações, existirem *data centers* heterogêneos [8].

Neste capítulo focamos em *data centers* com larguras de banda *Ethernet* heterogêneas e com SLAs restritivos no provisionamento de largura de banda para máquinas virtuais. Focamos no impacto da largura de banda de servidores na migração e desempenho de máquinas virtuais, considerando uma visão simplificada do *data center* como *single-hop star*, com um *switch* centralizado onde servidores, com diferentes interfaces de largura de banda *Ethernet*, estão conectados. Como uma ilustração simples, considere a Figura 5.1 como um cenário representativo do nosso trabalho. Aqui, $sw0$ é o *switch* central da topologia ao qual os servidores $h0$–$h3$ estão conectados com interfaces de diferentes larguras de banda. Esta topologia exemplifica um *switch top-of-rack* ao qual são conectados servidores de uma típica topologia multinível composta de *switches core*, *end-of-row* e *top-of-rack* em um *data center*.

Nossa principal contribuição apresentada neste capítulo é um novo método para o escalonamento de máquinas virtuais, capaz de ser executado em *data centers* homogêneos, mas especialmente projetado para *data centers* heterogêneos. Este método é constituído por dois algoritmos: (i) um *FHA*, executado por um *broker*, que objetiva o uso estratégico da largura de banda dos servidores; e (ii) um algoritmo, executado pelos servidores, de

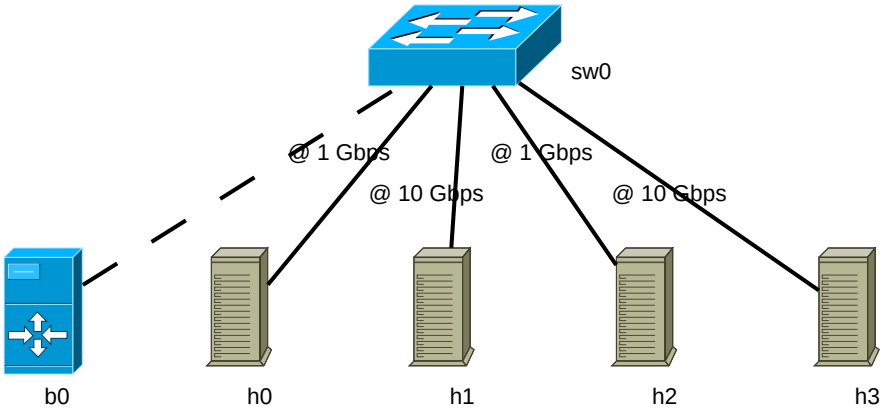

Figura 5.1: Visão Geral de um Cenário de Topologia

provisionamento de largura de banda — *Bandwidth Provisioning Algorithm* (*BPA*) — para máquinas virtuais.

Mais especificamente, nosso *FHA* objetiva tomar vantagem do fato que larguras de banda maiores podem potencialmente reduzir os tempos de migração de máquinas virtuais, e interagir, por comunicação através da rede, com um *BPA* capaz de reservar parte da largura de banda dos servidores para a migração de máquinas virtuais, em especial para máquinas virtuais previstas nos cenários deste capítulo — com SLAs restritivos em larguras de banda — e, como resultado, reduzir o consumo de energia. Nosso trabalho é significativo por dois caminhos: (i) no nosso conhecimento, esse trabalho é o primeiro a explorar um ambiente heterogêneo de rede para desenvolver um novo algoritmo; e (ii) nosso estudo sistemático mostra que nossa abordagem de fato traz economias significativas ao *data center*.

Especialmente neste capítulo, projetamos no método proposto um *FHA* para atuar conjuntamente ao *BPA*. Denominamos o *FHA* proposto como *Bandwidth-Aware Lago Allocator* (`BALA`), e apresentamos múltiplas opções de *BPA* com os quais ele pode atuar. Apresentamos no Capítulo 6 este *FHA* executado independentemente de um *BPA*, em cenários com SLAs não restritivos em larguras de banda. Portanto, para este capítulo, denominaremos como `BALA` o método constituído do *FHA* (`BALA`) juntamente ao *BPA* (`+E`) propostos.

Nós exemplificamos a interação mencionada na Figura 5.2: seja $b0$ um *broker*, executando um *FHA*; $h0$ e $h1$ são servidores, e cada um deles executa um *BPA* para uma máquina virtual. Quando $b0$ é ligado, ele obtém informações sobre todos os servidores em seu controle, incluindo CPU, RAM, largura de banda, etc. Quando é requisitado para $b0$ a designação de um servidor para uma máquina virtual ingressante, ele executa um *FHA*, que verifica entre todos os servidores qual deve receber a máquina virtual. Esta verificação pode ser feita através de informação previamente registrada em $b0$ sobre $h0$ e $h1$ ou através de uma comunicação ativa entre o *FHA* de $b0$ e os *BPAs* de $h0$ e $h1$. Quando $b0$ determinar qual servidor deverá receber a máquina virtual ingressante, e.g. $h1$, este executará algoritmos de provisionamento de recursos — CPU, RAM, largura de banda — para a máquina virtual ingressante e, entre estes algoritmos, um *BPA*. Agora, suponha que existe um cenário com um SLA restritivo, onde todas as requisições de lar-

gura de banda de todas as máquinas virtuais (para cada máquina virtual e sua respectiva migração) deva ser 100% suprido, e que existam duas máquinas virtuais alocadas em $h1$: $vm0$ e $vm1$. Note que esta configuração pode resultar em uma largura de banda livre e não usada em $h1$. O *BPA* pode agir com o objetivo de reservar esta largura de banda livre, que não poderia ser entregue à máquina virtual devido aos requerimentos de SLA, para migrações de máquinas virtuais. Isso tornaria migrações mais rápidas, permitindo um desligamento mais rápido de servidores que estejam ociosos em um contexto de ciência de energia.

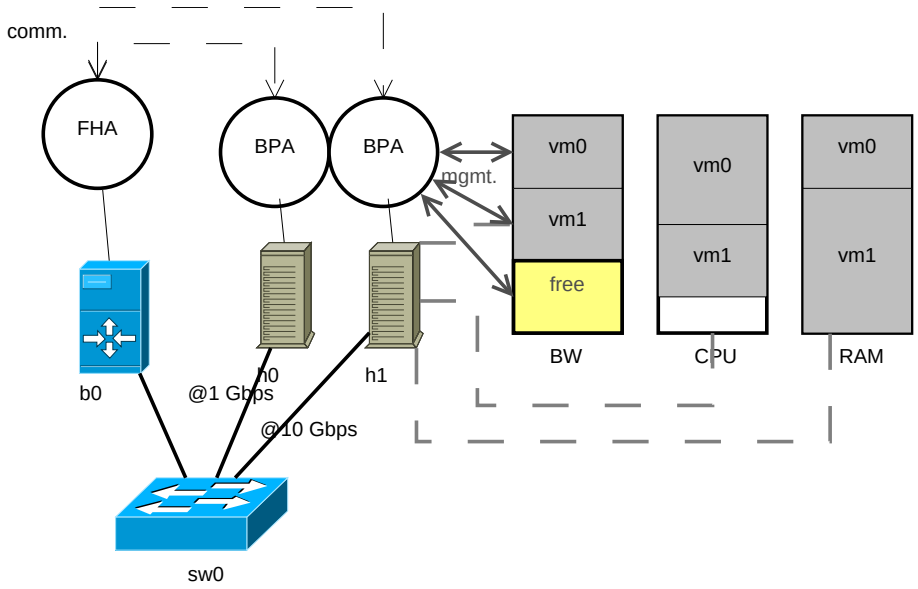

Figura 5.2: Interação entre o *FHA* e o *BPA*

Esta largura de banda extra a ser reservada para a migração de máquinas virtuais pode estimada por um *BPA*. Primeiramente, o *BPA* analisa quantas máquinas virtuais o servidor em questão ainda pode suportar, qual é o total de largura de banda que precisa ser garantida para estas máquinas virtuais baseado nos requerimentos de SLA, e realizar operações com estes valores para determinar se é possível prover largura de banda extra para a migração. Se este provisionamento for possível, a reserva é feita.

Os algoritmos que participam do método proposto agem integradamente para minimizar o consumo de energia. Apresentamos o *FHA* na Seção 5.1 e uma possível implementação de um *BPA* de provimento de banda adicional para este método na Seção 5.2. Os algoritmos são desenvolvidos com os seguintes princípios orientadores:

1. Cada servidor, sabendo os possíveis tipos de máquinas virtuais aos quais estão sujeitos a hospedar no *data center*, são capazes de calcular a largura de banda consolidada máxima que a soma das máquinas virtuais que poderão consumir/demandar;

2. Sabendo-se essa largura de banda, é possível verificar se haverá largura de banda disponível no servidor. Essa largura de banda poderá ser alocada como largura de banda extra (além da largura de banda base demandada pela máquina virtual) para ser usada, em conjunto com essa largura de banda base, para a migração das máquinas virtuais.

3. Esta largura de banda extra reservada pelo servidor não pode integrar requisições feitas pela máquina virtual, i.e. a largura de banda base demandada pela máquina virtual não pode ser aumentada, uma vez que ela pode ser migrada para um servidor com menos largura de banda, gerando um risco de violação de SLA;

4. Durante o processo de escolha de servidores para máquinas virtuais nós podemos tentar escolher servidores com larguras de banda maior, de modo a permitir potenciais migrações de máquinas virtuais mais rápidas e, portanto, acelerar o processo de desligamento de servidores ociosos, uma vez que eles ficarão livres mais rapidamente das máquinas virtuais migradas.

5.1 `BALA` — Encontrar Servidor para uma Máquina Virtual (*FHA*)

Nós enfatizamos que o objetivo do `BALA`, executado pelo *broker*, é reduzir o consumo de energia do *data center*, e para atingir este objetivo ele age em paralelo com o *BPA* apresentado na Subseção 5.2 para reduzir os tempos de migração. Esta redução é dada pelo fato que, por acelerando os tempos de migrações, existe uma liberação mais rápida do servidor de origem da máquina virtual, permitindo que ele se beneficie mais do DVFS, ou até mesmo seja desligado, caso fique ocioso.

Este *FHA* tem como escopo *data centers* homogêneos ou heterogêneos quanto à configuração de máquinas físicas. Além disso, ele não precisa de conhecimento prévio sobre peso das cargas de trabalho que serão alocadas nas máquinas virtuais do *data center*, pois ele verifica dinamicamente as necessidades de processamento e otimiza o consumo de energia *on-the-fly*.

Além disso, o *FHA* foi projetado para trabalhar bem com redes heterogêneas, constituídas de grupos de servidores com diferentes taxas de transmissão. No entanto, ele também pode ser aplicado a redes homogêneas, onde todos os servidores apresentam as mesmas taxas de transmissão na rede.

Agora descrevemos as funções utilizadas pelo *FHA*.

A função `getCpuUsage(`*host*`)` retorna o atual percentual de uso da CPU, $host_{mips}$ é o MIPS total que a CPU suporta e $host_{vms_{mips}}$ retorna a soma de MIPS de todas as máquinas virtuais alocadas no servidor, conforme mostrado na Equação (5.1).

$$\texttt{getCpuUsage(}host\texttt{)} = \frac{host_{vms_{mips}}}{host_{mips}} \tag{5.1}$$

Note que $host_{mips}$ reflete o desempenho efetivo de um processador. A fórmula padrão para calcular o número de Instruções Por Segundo (IPS) depende da frequência do *clock* e do número de Ciclos Por Instrução (CPI), como mostrado na Equação (5.2). No entanto, especialmente em arquiteturas computacionais modernas, o CPI pode variar para cada tipo de instrução. Então, uma técnica possível para medir este desempenho é a utilização de *benchmarkings* sintéticos, sendo um comumente utilizado o *Dhrystone* [126].

$$IPS = \frac{Frequency}{CPI} \tag{5.2}$$

A função `getPowerAfterAllocation` retorna a potência consumida de um servidor *host* após a alocação de uma máquina virtual *vm*. Em um modelo de potência linear, esta função pode ser definida como na Equação (5.3), onde $host_{max\_pow}$ é a potência máxima consumida por um servidor, composta por sua potência estática $host_{st\_pow}$ (constituída pelo processador ocioso, placa mãe, discos, RAM, adaptadores de rede, etc.) somada com uma potência dinâmica, que varia de acordo com a carga do servidor, portanto, dependente dos MIPS em uso atualmente pelo servidor e dos MIPS requeridos pela máquina virtual a qual deseja se alocar. $host_{used\_mips}$ representa a quantidade de MIPS de *host* em uso, e vm_{mips} representa a quantidade de MIPS solicitada por *vm*.

$$\texttt{getPowerAfterAllocation}(host,\ vm) = host_{st\_pow} + (host_{max\_pow} - host_{st\_pow}) \times \frac{host_{used\_mips} + vm_{mips}}{host_{mips}} \tag{5.3}$$

A função `bwForVm`(*host*, *vm*) retorna a largura de banda garantida que uma máquina virtual pode receber do servidor. A largura de banda mínima alocável para uma máquina virtual é a largura de banda estritamente requerida pela máquina virtual, mas o `BALA` explora esta função para tentar prover largura de banda extra para a migração de máquinas virtuais. Este recurso é apresentado com detalhes na Subseção 5.2.

A função `BALA findHostForVm`(*vm*) (Algoritmo 1) é executada pelo *broker* quando ele deseja determinar o servidor de destino de uma máquina virtual ingressante a ser instanciada ou migrada. Inicialmente (2), o algoritmo designa um valor menor ou igual a zero (e.g. infinito negativo) à variável $bestEfficiency$, que representa o melhor servidor em eficiência em energia encontrado no *data center*. Para calcular o valor desta variável, nós usamos (5.4), e quanto maior o valor, melhor. Após isso, a variável $bestHost$, que representa o melhor servidor candidato para a submissão da máquina virtual é inicializado com o valor `null`, que indica que inicialmente — antes do algoritmo analisar os servidores do *data center* — nenhum servidor é capaz de hospedar a máquina virtual. Finalmente, a variável $powerAtBestHostWithVm$, que representa a potência consumida estimada pelo servidor mais eficiente em energia com a máquina virtual a ser submetida, é inicializada com um valor infinito positivo.

A seguir, o algoritmo começa a varrer por características em cada servidor do *data center* para determinar se ele é adequado para uma máquina virtual ingressante (Linha 5), ou seja, se o servidor é capaz de suprir as demandas de recursos da máquina virtual em questão, como MIPS, RAM, etc.

Para cada servidor capaz de suportar a máquina virtual, o primeiro passo é o cálculo da eficiência em energia (Linha 7) como na Equação (5.4). Se a eficiência em energia encontrada do servidor sendo verificado é maior do que a eficiência em energia do servidor encontrado até agora, então o melhor servidor passa a ser considerado o melhor servidor, atualizando variáveis $bestHost$ e $bestEfficiency$ com os valores encontrados (Linhas 8–9).

Algoritmo 1 Bandwidth-Aware Lago Allocator: `findHostForVm(`vm`)`

```
1:  function findHostForVm(vm)
2:      bestEfficiency ← −∞
3:      bestHost ← null
4:      powerAtBestHostWithVm ← ∞
5:      for all capable hosts for vm in data center do
6:          efficiency ← host_mips / host_max_pow
7:          if efficiency > bestEfficiency then
8:              bestEfficiency ← efficiency
9:              bestHost ← host
10:         else if efficiency = bestEfficiency then
11:             powerAtHostWithVm ← getPowerAfterAllocation(host, vm)
12:             if powerAtHostWithVm < powerAtBestHostWithVm then
13:                 bestHost ← host
14:                 powerAtBestHostWithVm ← powerAtHostWithVm
15:             else if powerAtHostWithVm = powerAtBestHostWithVm then
16:                 if getCpuUsage(host) > getCpuUsage(bestHost) then
17:                     bestHost ← host
18:                 else if getCpuUsage(host) = getCpuUsage(bestHost) then
19:                     if host_mips > bestHost_mips then
20:                         bestHost ← host
21:                     else if host_mips = bestHost_mips then
22:                         if bwForVm(host, vm) > bwForVm(bestHost, vm) then
23:                             bestHost ← host
24:                         end if
25:                     end if
26:                 end if
27:             end if
28:         end if
29:     end for
30:     return bestHost
31: end function
```

$$EnergyEfficiency = \frac{host_{mips}}{host_{max\_pow}} \tag{5.4}$$

Na ocorrência de um empate no último critério, o que pode ser especialmente comum entre servidores de um mesmo grupo de hardware, uma decisão é realizada (Linha 10) através de outra comparação, estimando qual servidor possuirá o menor consumo de potência após a alocação da máquina virtual. Apesar dos servidores pertencerem a um mesmo grupo, é comum existirem resultados diferentes para esta comparação, em especial quando o recurso de DVFS estiver habilitado.

Caso ocorra outro empate, será verificado o nível de utilização do servidor (Linha 16). O servidor que possuir o nível de utilização mais alto será escolhido, uma vez que este tende a receber um maior número de máquinas virtuais e, potencialmente, demorar mais para desligar, evitando a migração das máquinas virtuais e que outros servidores fiquem

subutilizados. Observamos que a utilização desta estratégia pode, dependendo dos SLAs demandados pelas máquinas virtuais, gerar alguma tendência sobrecarga nos servidores, no sentido deles não serem capazes de prover a totalidade dos MIPS solicitados pelas máquinas virtuais que hospedam. Portanto, essa decisão estratégia implica num *trade-off* entre desempenho e eficiência em energia, uma vez que apesar de se reduzir o desempenho de máquinas virtuais que não apresentam demanda rígida de processamento mínimo, evita-se ligar novos servidores, reduzindo, desta maneira, o consumo consolidado de energia.

Se ainda existir um empate no último item, o que é comum se existirem muitos servidores que não estejam processando nenhuma máquina virtual, o servidor que possuir o maior poder de processamento será selecionado (Linha 19). Esta escolha é feita devido ao fato do servidor em questão tender a suportar maior número de máquinas virtuais.

Em caso de empate no último critério, nós verificamos qual é a largura de banda que o servidor pode prover para a máquina virtual (Linha 22). Este teste compreende em especial *data centers* que contém grupos de servidores com hardware semelhante, mas com diferentes larguras de banda, e tende a colocar um grande número de máquinas virtuais em servidores com maior largura de banda. Como uma máquina virtual pode já ter requerimento obrigatório de largura de banda, o objetivo desta escolha não é simplesmente "prover mais largura de banda para a máquina virtual". Como nós veremos na Subseção 5.2, esta escolha, em combo com o método de provisionamento de largura de banda para máquinas virtuais, possibilita a reserva de largura de banda adicional no servidor para permitir uma aceleração das migrações, caso venham a ocorrer. Em outras palavras, a migração de máquina virtual ocorrerá não só pela largura de banda base reservada à máquina virtual, mas com uma largura de banda extra — reservada para este propósito — do servidor, que dificilmente seria utilizada para outra finalidade.

Note que o bloco iniciado na Linha 22 é uma parte modificada do algoritmo `LA` original, e adicionamos este critério de desempate na Linha 22, ou seja, o nível mais profundo de critério de desempate. Este critério foi determinado após a realização de numerosos testes de simulação, de modo que foi encontrado que esta é a melhor posição para este critério de modo a reduzir o consumo de energia.

O Algoritmo 1, executado para cada máquina virtual ingressante, apresenta complexidade linear, O($hosts_{num}$), onde $hosts_{num}$ é o número de servidores que compõem a nuvem, dado em particular pela iteração da Linha 5. Considerando esta complexidade, para um número de máquinas virtuais vms_{num}, então a complexidade será $O(hosts_{num} \times vms_{num})$. Em cenários onde se aplica a migração de máquinas virtuais, usando $migrations_{num}$ como o número de migrações, então a complexidade total do processo se torna $O(hosts_{num} \times vms_{num} + migrations_{num})$.

A função `bwForVm (`*host*`, `*vm*`)` pode ser heuristicamente substituída pela `getBw(`*host*`)`, que retorna o total de largura de banda do servidor. Tal substituição implica que o servidor a ser escolhido é simplesmente aquele que possui maior largura de banda, ignorando a largura de banda que ele seria capaz de prover à máquina virtual. Isso é útil, por exemplo, para adaptar o Algoritmo 1 para casos onde não é possível conhecer os tipos de máquinas virtuais alocáveis, ou quando é sabido que todas as máquinas virtuais apresentam as mesmas demandas de largura de banda (ex: todas as máquinas virtuais requerem

100 Mbps), ou quando desejar tornar o `BALA` mais genérico e menos dependente do método de provisionamento de largura de banda para a máquina virtual.

Se for possível estimar os tamanhos das cargas de trabalho, uma outra otimização também pode ser feita. Considere, por exemplo, que uma máquina virtual está quase terminando o processamento de uma carga de trabalho. Migrar esta máquina virtual pode gerar desperdício de energia — seria necessário abortar a migração, pois o processamento da carga de trabalho terminaria antes da migração ser acabada. Para evitar isso, a Equação (2.2) pode ser utilizada para otimizar este algoritmo, adicionando uma estimativa da quantidade de energia que seria requerida pela máquina virtual, e realizar a migração somente se esta energia consumida for menor do que a energia para processar a máquina virtual no servidor atual. No entanto, o `BALA` tem como escopo primário cargas de trabalho de tamanhos sem padrão e, portanto, esta otimização não é aplicada.

Se todas as máquinas virtuais requisitadas não puderem ser alocadas, as máquinas virtuais restantes serão verificadas no próximo processamento do *broker*. À medida que as máquinas virtuais vão sendo instanciadas, as cargas de trabalho são submetidas para processamento; e à medida que estes processamentos vão sendo finalizados o *broker* tentará enviar as máquinas virtuais ainda não submetidas, migrar máquinas virtuais e, na existência de servidores ociosos, desligá-los.

5.2 *BPA* — O Algoritmo para Provisionamento de Largura de Banda para Máquinas Virtuais

No processo de escalonamento de máquinas virtuais pode existir, em adição ao *FHA*, executado pelo *broker*, um algoritmo responsável pelo provisionamento de largura de banda às máquinas virtuais (*BPA — Bandwidth Provisioning Algorithm*), executado pelos servidores. Nós chamaremos o algoritmo de provisionamento de largura de banda padrão como `SBPA` (*Standard Bandwidth Provisioning Algorithm*), responsável por alocar a banda base requisitada pro máquinas virtuais no servidor.

Nesta subseção, nós apresentamos um algoritmo de provisionamento de largura de banda estendido (*EBPA — Extended Bandwidth Provisioning Algorithm*), capaz de prover uma largura de banda extra, adicional à largura de banda requerida pelas máquinas virtuais, que é executado em paralelo com um *FHA*. Neste trabalho, esta largura de banda extra é reservada no servidor para fins de migração de máquinas virtuais. Além disso, nós exemplificamos o impacto do provisionamento de largura de banda realizado pelo *EBPA* nos tempos de migração.

Inicialmente, nós vamos considerar alguns princípios básicos, que devem ser considerados pelo *EBPA*, sobre a largura de banda reservável para uma máquina virtual ingressante. O primeiro princípio é que o algoritmo de provisionamento de largura de banda em um servidor deve ser capaz de atender as demandas da máquina virtual ingressante. Um outro princípio é que as reservas realizadas por este método não devem ser grandes demais, pois isso poderá fazer com que um servidor necessite hospedar um número reduzido de máquinas virtuais em decorrência da reserva excessiva de largura de banda.

Em um *EBPA* igualitário `+E`, cada servidor calcula uma largura de banda extra fixa, e este valor é dado para cada máquina virtual alocada no servidor. Para ilustrar a operação de uma possível implementação deste mecanismo de provisionamento de largura de banda extra para a migração de máquinas virtuais, considere um pequeno cenário composto por um servidor que suporte somente três máquinas virtuais, e estas três máquinas virtuais de diferentes tipos. Conhecendo os tipos de máquinas virtuais que podem ser alocadas, nós podemos estimar o consumo de largura de banda delas. Portanto, é possível reservar uma largura de banda extra igual para cada uma destas máquinas virtuais, conforme mostrado na Figura 5.3, onde *bbw* representa a largura de banda base requerida pela máquina virtual, e *ebw* refere-se à largura de banda extra reservada de modo igualitário.

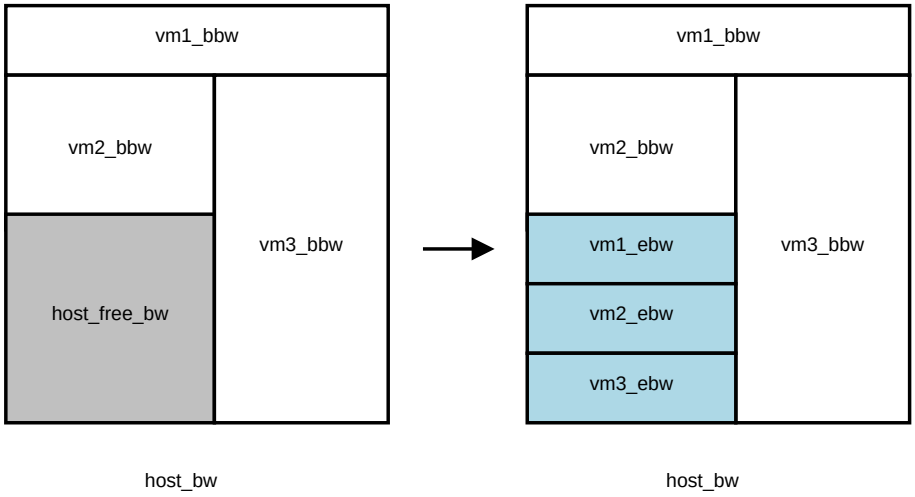

Figura 5.3: Reserva de Largura de Banda Estendida: Algoritmo Igualitário

O valor a ser dado para cada máquina virtual não deve exceder um limite superior, pois ele pode tornar o servidor inapto para receber novas máquinas virtuais devido à escassez de largura de banda. Conhecendo-se todos os tipos de máquinas virtuais, é possível calcular um bom valor para `+E`, dado pela Equação (5.5), onde $host_{vm\_extra\_bw}$ é a largura de banda extra que cada máquina virtual receberá do *host* em questão, $host_{bw}$ é a largura de banda total do servidor que receberá a máquina virtual, $host_{max\_vms}$ indica o maior número possível de máquinas virtuais suportadas pelo servidor, e $host_{max\_vms\_bw}$ indica a demanda de largura de banda base máxima consolidada requerida do servidor pelo maior grupo de máquinas virtuais que ele suporta.

$$host_{vm\_extra\_bw} = \frac{host_{bw} - host_{max\_vms\_bw}}{host_{max\_vms}} \tag{5.5}$$

Neste caso, $host_{max\_vms}$ é um problema reduzível ao *Bin Packing* 4-dimensional — um servidor pode suportar um número de máquinas virtuais considerando pelo menos o MIPS, RAM, largura de banda e armazenamento. Apesar ser ser um problema NP-difícil, usualmente o número de tipos de máquinas virtuais é pequeno, então a solução ótima $host_{optimal\_max\_vms}$ pode ser calculada computacionalmente através de força bruta, o que é desejável para que o `+E` opere no melhor caso. Por exemplo, considere dois tipos de máquinas virtuais: Tipo 1 demandando 2.000 MIPS, 0,5 GiB de RAM e 50 Mbps de

largura de banda; e Tipo 2 demandando 1.500 MIPS, 1 GiB de RAM e 50 Mbps; em um cenário com SLA garantindo todas as demandas. Considere que ambos tipos possuam requisitos de armazenamento desprezíveis. Considere um servidor que possui 6.000 MIPS, 8 GiB de RAM e 1 Gbps de largura de banda. Portanto, o número máximo de máquinas virtuais suportadas por este servidor será 4, descoberto por força bruta (um servidor pode suportar até 4 máquinas virtuais do Tipo 2), sendo MIPS o fator limitador de modo que cada servidor conseguirá, no seu melhor, suportar 4 máquinas virtuais (Tipo 2, cada uma requerendo 1.500 MIPS) executadas em paralelo. Com conhecimento prévio de quais tipos de máquinas virtuais podem ser alocada no *data center*, e de quais máquinas virtuais estão atualmente alocadas em um dado servidor, é possível estimar quais máquinas virtuais ainda poderão ser alocadas neste servidor. Neste exemplo, se um servidor já possui uma máquina virtual do Tipo 2 alocada, então ele terá disponível 4.500 MIPS e 7 GiB de RAM. Nesta situação, ainda será possível alocar 2 máquinas virtuais do Tipo 1, ou 3 máquinas virtuais do Tipo 2, ou um misto de máquinas virtuais — 1 do Tipo 1 e 1 do Tipo 2. Além disso, não existe um problema de quebra do algoritmo se $host_{max\_vms\_bw}$ for estimado acima do valor real. No entanto, haverá uma consequência importante: o provisionamento de largura de banda extra não será feito de modo ótimo.

O cálculo do $host_{max\_vms\_bw}$ também é reduzível ao problema do *Bin Packing*, uma vez que existem diversos tipos de máquinas virtuais, cada tipo com seu próprio requerimento de hardware, tornando este cálculo complexo. No entanto, se nós considerarmos todos os tipos de máquinas virtuais possuindo os mesmos requerimentos de largura de banda, o calculo deste valor poderá ser feito de modo simples segundo a Equação (5.6), onde vm_{bw} retorna a quantidade de largura de banda requerida pela máquina virtual especificada. Consideraremos neste capítulo que todos os tipos de máquinas virtuais possuem as mesmas demandas de largura de banda.

$$host_{max\_vms\_bw} = host_{max\_vms} \times vm_{bw} \tag{5.6}$$

Considerando o exemplo anterior, nós podemos usar os valores de $host_{max\_vms}$ encontrados aplicados na Equação (5.5), desenvolvendo (Equação (5.7)) e encontrando que, neste caso, para cada máquina virtual alocada em *host*, nós podemos reservar uma largura de banda extra de 200 Mbps, além da largura de banda base de 50 Mbps já demandada por cada máquina virtual, para acelerar o processo de migração das máquinas virtuais. Note que, neste caso, mesmo em um *data center* completamente homogêneo — com todos os servidores tendo a mesma largura de banda — nós temos um *speedup* de 5 vezes no tempo de migração das máquinas virtuais. E, se o servidor possuísse uma largura de banda de 10 Gbps, cada máquina virtual poderia receber uma largura de banda extra de 2,45 Gbps para migrações, conforme mostrado na Equação (5.8).

$$host_{vm\_extra\_bw} = \frac{1.000 - 4 \times 50}{4} = 200 \tag{5.7}$$

$$host_{vm\_extra\_bw} = \frac{10.000 - 4 \times 50}{4} = 2.450 \tag{5.8}$$

Com base no apresentado, nós podemos definir a função `bwForVm`, empregada no Algoritmo 5.1 de acordo com a Equação (5.9).

$$\texttt{bwForVm}(host,\ vm) = host_{vm\_extra\_bw} + vm_{bw} \tag{5.9}$$

Um problema potencial no cálculo de $host_{max\_vms\_bw}$ (Equação (5.5)) é a obrigatoriedade de se calcular $host_{max\_vms}$, que por sua vez requer conhecimento prévio das demandas de hardware de todas as máquinas virtuais alocáveis. Isto pode não ser prático em *data centers* que possuem alta elasticidade quanto aos tipos de máquinas virtuais alocáveis — i.e., possuindo um grande número de tipos de máquinas virtuais. Um caminho possível de lidar com isso, com o objetivo de evitar possuir um mal resultado no provisionamento de largura de banda extra para as máquinas virtuais, é impôr restrições na instanciação de máquinas virtuais entre os servidores com o objetivo de determinar com uma função quantas máquinas virtuais $host_{max\_vms}$ um servidor pode alocar, e garantir que todos os tipos de máquinas virtuais possuam larguras de banda iguais e um conjunto de parâmetros bem conhecido e facilmente trabalhável. Se estas condições são satisfeitas, é possível calcular, com uma aceitável margem de erro, a largura de banda extra para uma máquina virtual usando `+E` com a Equação (5.10).

$$host_{vm\_extra\_bw} = \frac{host_{bw} - host_{max\_vms} \times vm_{bw}}{host_{max\_vms}} \tag{5.10}$$

Se a heurística que substitui a função $\texttt{bwForVm}(host,\ vm)$ pela função $\texttt{getBw}(host)$ for usada, então a largura de banda extra a ser reservada em uma distribuição igualitária poderá também ser calculada pela Equação (5.10).

O desempenho de largura de banda para as máquinas virtuais devido ao provisionamento extra depende das garantias previstas em SLAs. Em outras palavras, quanto menor é a garantia de largura de banda para máquinas virtuais, maior é a largura de banda que um *EBPA* poderá reservar para acelerar as migrações. Por outro lado, em cenários restritivos que garantem altos valores de largura de banda para máquinas virtuais, menor será o desempenho do *EBPA*.

Além do método de provisionamento para largura de banda extra `+E`, nós também podemos fazer uso de outros métodos de *EBPA*, como por exemplo, um método diferenciado. Diferentemente do mecanismo igualitário, onde todas as máquinas virtuais alocadas em um servidor recebem larguras de banda iguais para a realização de migração, o método diferenciado possibilita que determinadas máquinas virtuais, alocadas em um mesmo servidor, recebam diferentes larguras de banda extra entre si. Isto pode ser uma vantagem, pois podemos tentar prover mais largura de banda para máquinas virtuais que potencialmente migrarão mais (e.g. máquinas virtuais com cargas de trabalho mais pesadas), enquanto pode ser também uma desvantagem, pois se uma má decisão for feita, máquinas virtuais que recebem pouca largura de banda extra terão seus tempos de migração prejudicadas em relação àquelas que receberam mais largura de banda. Não existe um algoritmo padrão definido para o mecanismo diferenciado: de fato este é um método

de designação personalizado, que pode ser adaptado pelo administrador da nuvem com estratégias apropriadas à nuvem, o que pode fazer este método complexo.

Realizando uma análise mais profunda, em especial no caso de cenários possuindo configurações heterogêneas de máquinas virtuais, nós observamos que o método de provisionamento igualitário de largura de banda `+E` pode ter algumas deficiências. Para ilustrar isso, considere um cenário de um *data center* que apresenta dois grupos de servidores diferenciados somente pela largura de banda: o Grupo I possui servidores com largura de banda de 1 Gbps; enquanto a largura de banda para cada servidor do Grupo II é de 10 Gbps. Considere, ainda, dois tipos de máquinas virtuais com exigências de largura de banda em SLA: o Tipo A requer 25 Mbps de largura de banda; enquanto o Tipo B requer 250 Mbps de largura de banda. Neste cenário, desconsiderando a largura de banda, cada servidor suporta no máximo 10 máquinas virtuais. Portanto, encontramos que o maior número de máquinas virtuais alocáveis $host_{max\_vms}$ será de 10 máquinas virtuais Tipo A para cada servidor do Grupo I; 10 máquinas virtuais Tipo A para cada servidor do Grupo II; 4 máquinas virtuais Tipo B para cada servidor do Grupo I; e 10 máquinas virtuais Tipo B para cada servidor do Grupo II.

Após a implementação do método igualitário nosso passo é calcular $host_{vm\_extra\_bw}$ com a Equação (5.5) para servidores dos grupos I e II. Para calcular $host_{vm\_extra\_bw}$, nós precisamos encontrar alguns valores: $host_{bw}$ é uma informação dada – para o Grupo I é 1 Gbps e para o Grupo II é 10 Gbps. Considerando que o número de máquinas virtuais seja elevado, podemos considerar $host_{max\_vms\_bw}$ como sendo a largura de banda consolidada máxima consumida pelas máquinas virtuais que mais consomem largura de banda, neste caso Tipo B — portanto, esta quantidade é $host_{max\_vms\_bw} = 10VMs \times 250$ Mbps$= 2,5$ Gbps para servidores do Grupo II, enquanto é $host_{max\_vms\_bw} = 4VMs \times 250$ Mbps$=$ 1 Gbps para servidores do Grupo I; e $host_{max\_vms}$ já foi calculado. Portanto, temos que $host_{vm\_extra\_bw} = (10$ Gbps-2.5 Gbps$)/10 = 750$ Mbps para cada servidor do Grupo II e $host_{vm\_extra\_bw} = (1$ Gbps-1 Gbps$)/4 = 0$ Mbps para cada servidor do Grupo I.

Agora, baseados na Equação (5.11), que define o aproveitamento mínimo de banda $BwHarnessing$ para um *host* totalmente carregado, para máquinas virtuais que utilizam SLAs restritivos, onde $lowestBwVm$ representa o tipo de máquina virtual alocável que consome a menor quantidade de largura de banda, nós observamos que para o método igualitário o aproveitamento é de $((25 + 750) \times 10)/(10$ Gbps$) = 77.5\%$ para servidores do Grupo II e somente $((25 + 0) \times 4)/(1$ Gbps$) = 10\%$ para servidores do Grupo I. Note, portanto, que apesar do método igualitário funcionar bem em grupos de máquinas virtuais com configurações similares de largura de banda, ele pode levar, em alguns cenários, a uma largura de banda não utilizada — desperdiçada — e, como mostrado neste cenário, a largura de banda desperdiçada pode chegar a $100\% - 10\% = 90\%$.

$$BwHarnessing_{host} = \frac{\texttt{bwForVm}(host,\ lowestBwVm) \times host_{max\_vms}}{host_{bw}} \tag{5.11}$$

Considerando as deficiências apresentadas pelo método igualitário, e o fato que o método diferenciado pode ser difícil de implementar devido às personalizações do ambiente onde ele será aplicado, nós idealizamos aqui um modelo híbrido/adaptativo de provisio-

Algoritmo 2 Bandwidth-Aware Lago Allocator: `adaptiveAllocateBwForVm(`vm`)`

1: **procedure** ADAPTIVEALLOCATEBWFORVM(vm)
2: $host_{max\_remaining\_vms} \leftarrow$ `calcBinPackingMaxRemainingVMsForHost(`$allVMs$`)`
3: $host_{vm\_extra\_bw} = freeBwAfterAllocation_{vm}/host_{max\_remaining\_vms}$
4: $bwToAllocate = vm_{bw} + host_{vm\_extra\_bw}$
5: `allocateBwForVM(`$bwToAllocate$`,` vm`)`
6: **end procedure**

namento. Este modelo pode atuar em cenários onde máquinas virtuais se comportam de modo próximo e a aplicação do método igualitário seria aceitável, mas é adaptativo, de modo a tentar lidar com futuras situações de provisões, sendo guiado por princípios de uma distribuição balanceada de largura de banda entre as máquinas virtuais alocáveis, objetivando maximizar a utilização dos servidores do *data center*, e aplicável a cenários genéricos. Nós apresentamos o Algoritmo 2, executável pelos servidores do *data center* para prover largura de banda para máquinas virtuais em cenários com SLAs restritivos em provisionamento de largura de banda para máquinas virtuais, onde nós identificamos por *host* o servidor atual executando o algoritmo e por *vm* a máquina virtual a ser alocada.

O Algoritmo 2 opera do seguinte modo: na Linha 2 o algoritmo calcula, considerando os tipos de máquinas virtuais restantes, entre todas as combinações de tipos de máquinas virtuais suportadas pelo servidor, quantas máquinas virtuais ainda podem ser alocadas, o que pode ser feito utilizando uma heurística de solução do problema *Bin Packing*. A melhor opção é encontrar o valor ótimo para este número de máquinas virtuais ainda suportadas $host_{max\_remaining\_vms}$, mas o algoritmo pode funcionar normalmente se a condição $host_{max\_remaining\_vms} \leq host_{optimal\_max\_remaining\_vms}$ for satisfeita, onde $host_{optimal\_max\_remaining\_vms}$ é a solução ótima para o problema do *Bin Packing*. Uma aproximação de qualidade questionável pode ser feita utilizando a Equação (5.12), onde $highestVm_{parameter}$ representa a máquina virtual $highestVm$ que requer a maior quantidade do recurso $parameter$, e $host_{remaining\_parameter}$ representa os recursos disponíveis de $parameter$ do servidor $host$. No próximo passo (Linha 3), o algoritmo estima quanto de largura banda extra deve ser alocado para a vm a ingressar, o que pode ser determinado pela razão entre largura de banda disponível no servidor após a alocação de vm e o número de máquinas virtuais que ainda poderão ser suportadas pelo servidor — como o servidor ainda não alocou vm, então este valor inclui as demandas de vm. Após isso, na Linha 4 o servidor calcula quanto de largura de banda consolidada, constituída pela banda base da máquina virtual — a largura de banda requerida pela máquina virtual — com a largura de banda extra — calculada na Linha 3 — ele reservará para vm. O último passo é o provisionamento de fato da largura de banda, realizada na Linha 5.

$$host_{max\_remaining\_vms} = max\left\{\frac{host_{remaining\_mips}}{highestVm_{mips}}, \frac{host_{remaining\_ram}}{highestVm_{ram}}, \frac{host_{remaining\_bw}}{highestVm_{bw}}, \frac{host_{remaining\_stor}}{highestVm_{stor}}\right\} \tag{5.12}$$

Note que o Algoritmo 2 garante 100% de aproveitamento da largura de banda do servidor. Isso pode ser provado da seguinte maneira: assuma que o servidor pode suportar somente a última máquina virtual, então o denominador da equação referenciada na Linha 3 é 1, então a largura de banda extra calculada será igual à largura de banda base demandada pela máquina virtual, acrescida de toda a largura de banda restante do servidor após a alocação, portanto a largura de banda a ser alocada, calculada em 4, deixará 0 Mbps livres após esta alocação, portanto garantindo uma utilização de 100%.

Observamos, contudo, que a atuação dos *BPAs* se dá apenas nos servidores, não contemplando o núcleo da rede. Embora as reservas realizadas nos servidores consigam garantir a largura de banda por parte destes, existe a possibilidade que o núcleo da rede — área fora do escopo de atuação dos *BPAs* — dependendo da topologia utilizada, não consiga atender 100% da demanda consolidada. Portanto, conhecendo as limitações das redes do *data center*, um administrador poderá fazer adequações específicas para cada caso, com o intuito de minimizar os impactos de tais limitações.

5.3 Ilustração

Na Figura 5.4 nós apresentamos um exemplo simplificado de um cenário operacional para o método de alocação proposto. Seja $h0$ um servidor Tipo 1, com largura de banda $h0\_bw$; e $h1$ um servidor Tipo 2, possuindo configuração idêntica a $h0$, exceto pela largura de banda. Estas larguras de banda estão representadas em uma escala não linear na figura. Seja $h1\_bw$ a largura de banda de $h1$, de modo que $h1\_bw = 10 \times h0\_bw$. Considere que todas as máquinas virtuais apresentam a mesma configuração, e que todos os SLAs sejam restritivos em largura de banda, i.e. uma largura de banda requerida por uma máquina virtual deve ser 100% satisfeita. Considere que o limite superior de máquinas virtuais hospedáveis por um servidor não seja dado em função da largura de banda.

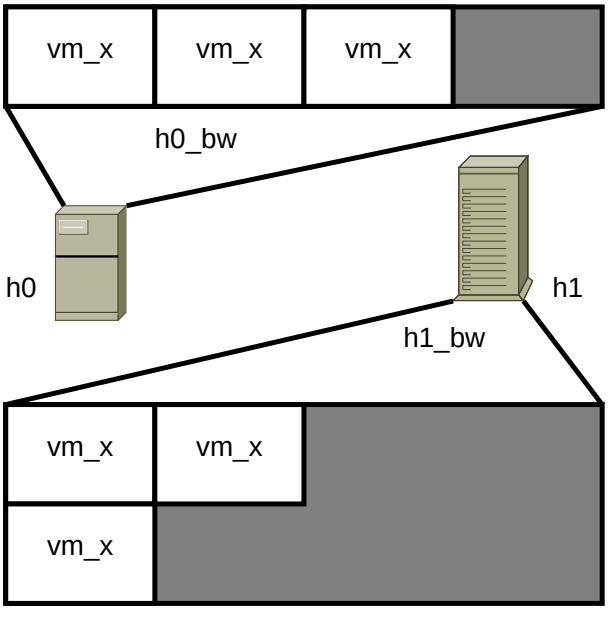

Figura 5.4: Cenário de Exemplo para o Bandwidth-Aware Lago Allocator

Notamos que neste cenário existe um desperdício de largura de banda em $h1$, uma vez que mesmo que todas as máquinas virtuais sejam alocadas juntamente em um servidor, ainda haverá muito de $h1\_bw$ não utilizado. Por exemplo, se o limite hospedagem em um servidor for de 3 máquinas virtuais, e cada máquina virtual consumir 250 Mbps de largura de banda, sendo $h0\_bw$ 1 Gbps e $h1\_bw$ 10 Gbps, então $h0\_bw$ será capaz de prover a largura de banda consolidada demandada pelas máquinas virtuais e restarão 250 Mbps. Supondo a alocação destas mesmas máquinas virtuais em $h1$, então haverão 9,25 Gbps não utilizados. Uma regra simples para permitir evitar este desperdício no segundo caso seria prover mais largura de banda para beneficiar a migração de máquinas virtuais, no entanto esta abordagem implica em dois problemas: (i) (um problema pequeno) a violação de SLA, pois estaríamos dando mais largura de banda do que o requisitado pelas máquinas virtuais; e (ii) uma vez que a largura de banda extra seja reservada, se tivermos que manter a largura de banda neste nível alto, nós impossibilitaríamos a migração de máquinas virtuais alocadas em $h1$ para $h0$, pois $h0\_bw$ seria insuficiente para suportar a as máquinas virtuais de $h1$, uma vez que a largura de banda reservada em $h1\_bw$ seria muito maior do que $h0\_bw$.

Agora consideramos uma máquina virtual Tipo 2, definida na Seção 5.2, e com uma quantidade de dados para migrar igual à RAM consumida pela máquina virtual, neste caso, 1 GB. Com o modelo de migração de máquina virtual utilizando a largura de banda base da máquina virtual em questão (neste caso 50 Mbps), o tempo consumido será de $(1.000\ MB)/(50\ Mbps) = (1.000{\times}8Mb)/(50\ Mbps) = 160$ s considerando o cenário ótimo. Se o método de migração proposto na Seção 5.2 for utilizado para o mesmo cenário, então o tempo consumido seria de aproximadamente $(1.000\ MB)/(50\ Mbps + 200\ Mbps) = (1.000 \times 8Mb)/(250\ Mbps) = 32$ s, o que representa, em outras palavras, uma redução de 80% no tempo de migração.

5.4 Simulação

Para analisar os benefícios do *FHA* `BALA` para encontrar servidores para máquinas virtuais, atuando juntamente ao *EBPA* apresentado, utilizamos o simulador *CloudSim*, realizando as seguintes modificações em seu núcleo para possibilitar os experimentos:

- Estendemos a classe `VmAllocationPolicy` para suportar os algoritmos *FHA* estudados nesta seção;
- Modificamos as classes `PowerDataCenter` e `PowerDataCenterNonPowerAware` para os algoritmos de escalonamento de máquinas virtuais poderem interagir com o algoritmo *BPA* estudado nesta seção.

Nas próximas subseções, nós apresentamos em detalhes como as simulações foram configuradas e executadas.

5.5 Configuração do *Data Center*

Assumindo que o número de máquinas virtuais é seis vezes o número de servidores, e que cada máquina virtual processa uma *cloudlet*, então o número de *cloudlets* é igual ao número de máquinas virtuais e, portanto, seis vezes o número de servidores. Apresentamos na Tabela 5.1 os tamanhos dos *data centers*, número de máquinas virtuais e tamanho das cargas de trabalho utilizados neste trabalho.

Tabela 5.1: Tamanhos de *Data Centers* e Cargas de Trabalho

Tamanho	# Servidores	# Máquinas Virtuais	Total de Cargas de Trabalho do *Data Center* (em Milhões de Instruções)
$s10$	10	60	122.760.000
$s100$	100	600	1.227.600.000
$s1000$	1.000	6.000	12.276.000.000

Nós agora comentamos sobre o número de máquinas virtuais por servidor e o relacionamento para as cargas de trabalho. Se um *data center* $s1000$ recebe um número muito pequeno de máquinas virtuais e cargas de trabalho para processar, ele poderá terminá-las rapidamente, praticamente não dando tempo para as migrações e não permitindo uma avaliação adequada do consumo de energia. Por outro lado, se um *data center* $s10$ recebe um número muito grande de máquinas virtuais e cargas de trabalho, o processamento poderá consumir tempo demais, também não permitindo uma avaliação adequada do consumo de energia. Em outras palavras, nós encontramos que é adequado que o número de máquinas virtuais e cargas de trabalho serem proporcionais ao tamanho do *data center*. Especificamente, para estes testes consideramos que uma boa regra é o número de máquinas virtuais ser seis vezes o número de servidores, e cada máquina virtual recebe uma *cloudlet* por vez.

Portanto, cargas de trabalho são submetidas às máquinas virtuais como *cloudlets*, de modo que cada *cloudlet* tenha 30.000. 120.000. 480.000. 1.920.000 e 7.680.000 milhões de instruções (MI) em distribuição *round-robin*. Por exemplo, considerando estes cinco tamanhos de *cloudlets*, em um *data center* $s10$ com 60 máquinas virtuais, teremos 12 máquinas virtuais processando 12 *cloudlets* de 30.000 MI cada, 12 máquinas virtuais processando 12 *cloudlets* de 120.000 MI, e assim por diante, seguindo o mecanismo *round-robin*. Portanto, o total de cargas de trabalho em um *data center* $s10$ será de 122.760.000 MI ($= 12 \times (30.000 + 120.000 + 480.000 + 1.920.000 + 7.680.000)$). De modo similar, os totais de cargas de trabalho determinadas para outras configurações de *data centers* estão apresentadas na Tabela 5.1.

Finalmente, nós criamos cenários com cada um destes *data centers*, operando com larguras de banda de 100 Mbps, 1 Gbps, 10 Gbps, 100 Gbps, 1 Tbps e 10 Tbps; e utilizando o método de provisionamento de banda estendida igualitário `+E`, apresentado na Subseção 5.2, para a migração de máquinas virtuais.

5.5.1 Configurações de Servidores e Máquinas Virtuais

A configuração comum para todos os servidores é: 8 GiB de RAM e 2,5 GB de tamanho de imagem. Os servidores são diferenciados nos adaptadores de redes, que apresentam taxas de transmissão definidas para 100 Mbps, 1 Gbps, 10 Gbps, 100 Gbps, 1 Tbps e 10 Tbps, em uma distribuição *round-robin*. Os valores do consumo de potência dos servidores são baseados em [106]. A Tabela 5.2 sumariza as frequências e consumo de potência para cada servidor para cada tipo de *data center*.

Tabela 5.2: Distribuição de Servidores para *Data Centers*

ID do Servidor	Parâmetro	Homogêneo	Misto	Heterogêneo
$h0$	Frequência (GHz)	4×3,06	4×3,06	4×1,6
	Potência (W)	250	200	200
$h1$	Frequência (GHz)	4×3,06	8×3	4×1,86
	Potência (W)	250	250	250
$h2$	Frequência (GHz)	4×3,06	4×3,06	4×3,06
	Potência (W)	250	300	300
$h3$	Frequência (GHz)	4×3,06	8×3	4×3,3
	Potência (W)	250	200	350
$h4$	Frequência (GHz)	4×3,06	4×3,06	8×3
	Potência (W)	250	250	200
$h5$	Frequência (GHz)	4×3,06	8×3	4×1,6
	Potência (W)	250	300	250

Com relação às máquinas virtuais, suas RAM foram definidas para 512 MiB. A largura de banda requisitada por cada máquina virtual é de 8,33 Mbps para garantir que a largura de banda não seja o limite superior de número de máquinas virtuais alocáveis em um servidor, mas também não pequeno demais para evitar que um grupo de máquinas virtuais utilize menos do que a largura de banda máxima garantida de qualquer servidor do *data center* para máquinas virtuais. As capacidades de processamento das máquinas virtuais foram definidas para 1.000, 2.000 e 3.000 MIPS em distribuição *round-robin*.

5.5.2 Configurações de *Data Center* Homogêneos

Para simular *data centers* homogêneos, para cada servidor foram escolhidos, uniformemente, o processador Intel Xeon X5667 (4 × 3,06 GHz), um consumo de 250 W.

5.5.3 Configurações de *Data Centers* Heterogêneos

Para simular *data centers* heterogêneos, as seguintes configurações foram adotadas:

- Processador → cada servidor possui um dos seguintes tipos de processadores, em uma distribuição *round-robin*: Intel Xeon E5603 (4 × 1,6 GHz), Intel Xeon E7520 (4 × 1,86 GHz), Intel Xeon X5667 (4 × 3,06 GHz), AMD Opteron 6204 (4 × 3,3 GHz) e AMD Opteron 6220 (8 × 3 GHz);

- Potência → cada servidor possui uma das seguintes potências, em uma distribuição *round-robin*: 200, 250, 300 e 350 W.

Ilustramos na Tabela 5.3 como os processadores e as potências dos servidores estão distribuídos neste tipo de *data center*.

Tabela 5.3: Exemplo de Configuração de Servidores em um *Data Center* Heterogêneo

Servidor	h_0	h_1	h_2	h_3	h_4	h_5
MIPS×Núcleos	1.600×4	1.860×4	3.066×4	3.300×4	3.000×8	1.600×4
Potência (W)	200	250	300	350	200	250

5.5.4 Configuração de *Data Centers* Mistos

Para simulações realizadas em *data centers* mistos, as seguintes configurações foram adotadas: cada servidor possui um dos dois tipos de processadores, em distribuição *round-robin*: Intel Xeon X5667 (4 × 3,06 GHz) e AMD Opteron 6220 (8 × 3 GHz). Quanto à potência dos servidores, também foi adotada uma distribuição *round-robin* de 200, 250 e 300 W.

Ilustramos na Tabela 5.4 como o MIPS/potência dos servidores são distribuídos neste tipo de *data center*.

Tabela 5.4: Exemplo de Configuração de Servidores em um *Data Center* Misto

Servidor	h_0	h_1	h_2	h_3	h_4	h_5
MIPS×Núcleos	3.066×4	3.000×8	3.066×4	3.000×8	3.066×4	3.000×8
Potência (W)	200	250	300	200	250	300

5.6 Regras de Simulação

Apesar dos *FHAs* clássicos `HU` e `MPD` atuarem como `SBPA`, nós realizamos modificações para que eles pudessem suportar o *EBPA* proposto neste trabalho, com a possibilidade de torná-los possivelmente melhores, para compará-los, de modo justo, em suas melhores configurações, com o `BALA`, com o objetivo de determinar a melhor opção a ser usada para todos os tipos de *data centers* para os cenários abordados. O *FHA round-robin* não foi contemplado, visto que seus resultados foram considerados ruins comparados aos demais algoritmos de escalonamento.

5.7 Resultados e Análise

Os principais critérios escolhidos para análise e comparação foram consumo de energia e *makespan*. Um bom algoritmo de escalonamento deve ser eficiente em energia, mas não deve consumir muito tempo para processar cargas de trabalho.

Nas Figuras 5.5–5.13, nós apresentamos o consumo de energia para cenários compostos de *data centers* homogêneos, mistos e heterogêneos, com tamanhos $s10$, $s100$ e $s1000$.

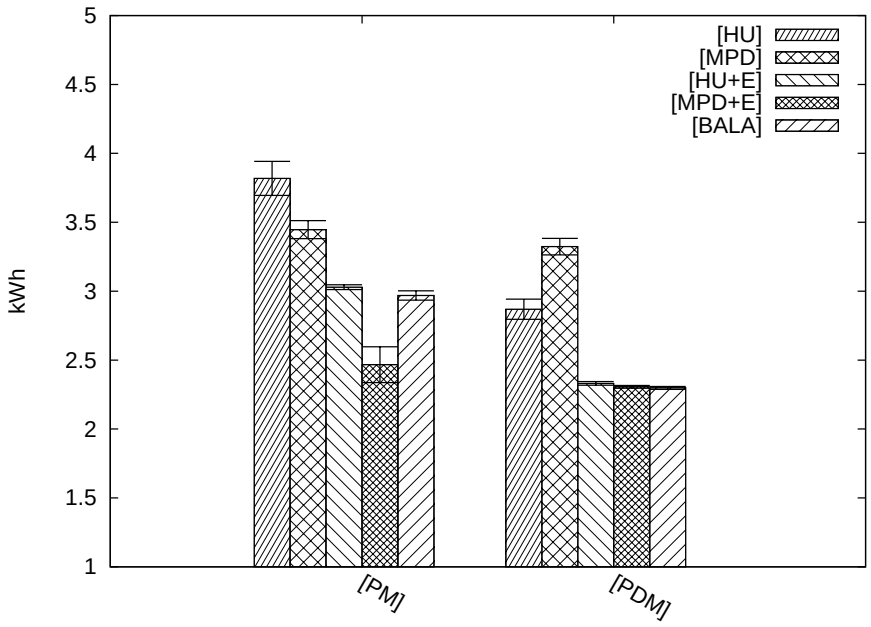

Figura 5.5: Consumo de Energia: *Data Center* Homogêneo, 10 Servidores

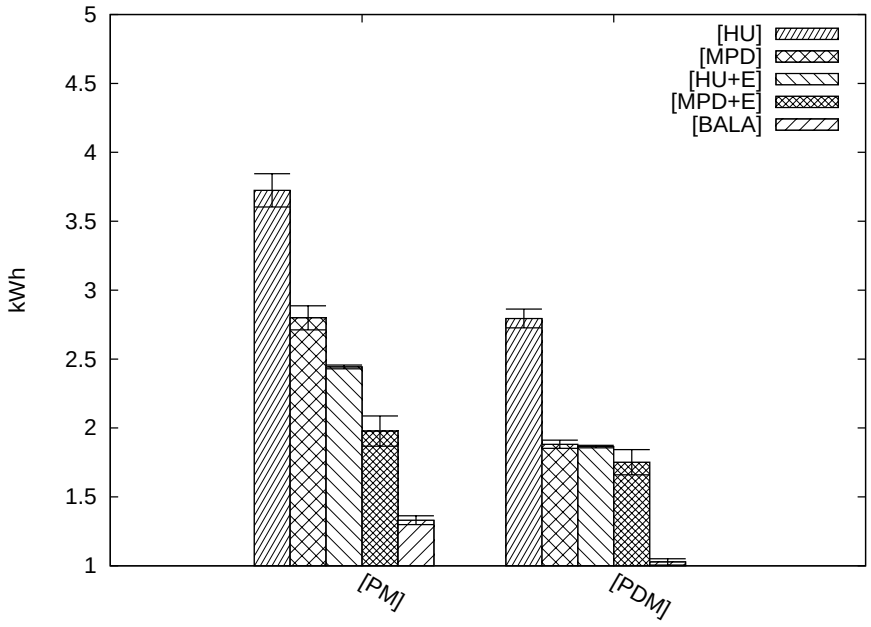

Figura 5.6: Consumo de Energia: *Data Center* Misto, 10 Servidores

Como podemos observar, a adição do método de provisionamento de largura de banda de máquinas virtuais proposto neste trabalho mostra significativa melhora para os *FHAs* `HU` e `MPD`. De fato, esta melhoria foi tão impactante que o `HU+E` superou o `MPD` na maioria dos casos, e.g. nos *data centers* homogêneos o `HU+E-PDM` foi em média 43% mais eficiente em energia que o `MPD-PDM`.

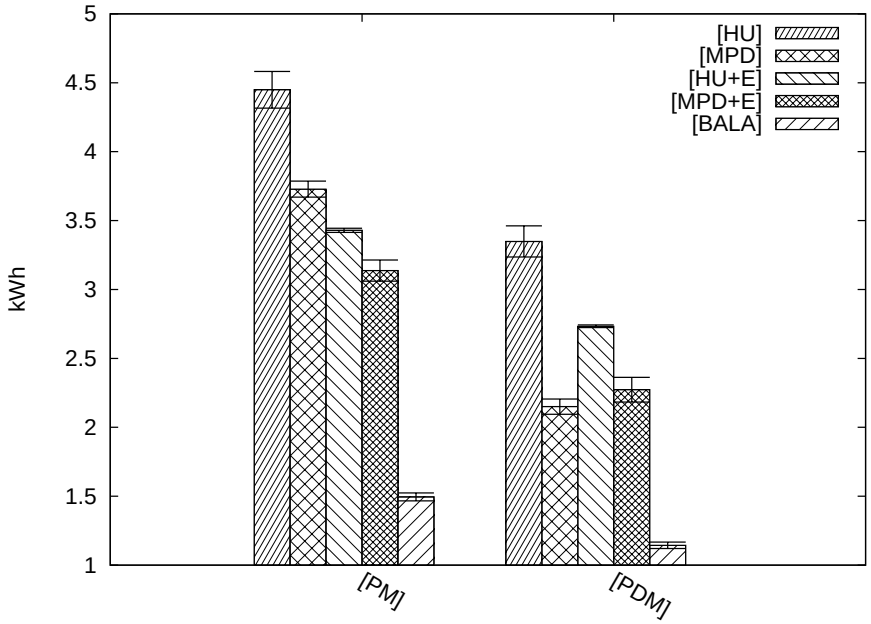

Figura 5.7: Consumo de Energia: *Data Center* Heterogêneo, 10 Servidores

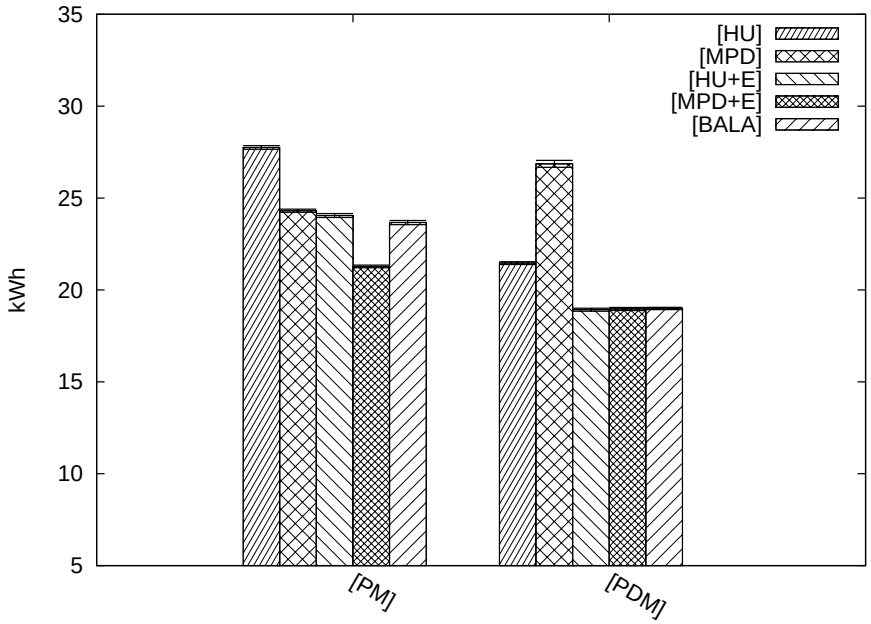

Figura 5.8: Consumo de Energia: *Data Center* Homogêneo, 100 Servidores

Além disso, considerando a configuração `PM`, o `BALA` se mostrou o melhor em todos os casos de *data centers* mistos e heterogêneos, atrás do `MPD+E` somente em *data centers* completamente homogêneos. Considerando a configuração `PDM`, a qual — do ponto de vista de eficiência em energia — é a mais indicada na prática, tal perda de fato não ocorre, e nos casos de *data centers* homogêneos $s1000$, o `BALA-PDM` desempenha melhor do que o `MPD+E-PDM` em cerca de 1,7%.

Com relação ao *makespan*, notamos pequenas diferenças. Em alguns casos, os *FHAs* com adição do recurso `+E` consumiram mais tempo, em outros este tempo foi reduzido.

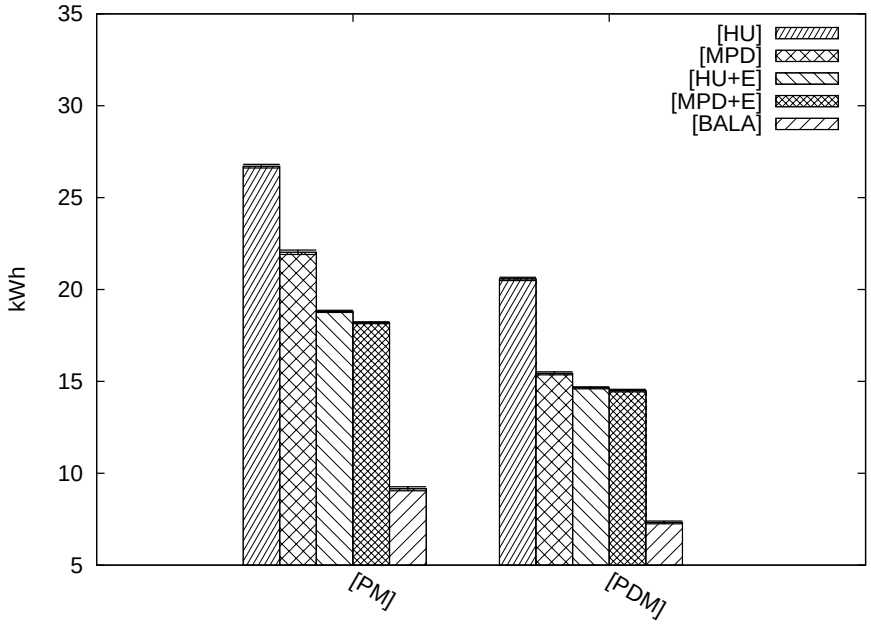

Figura 5.9: Consumo de Energia: *Data Center* Misto, 100 Servidores

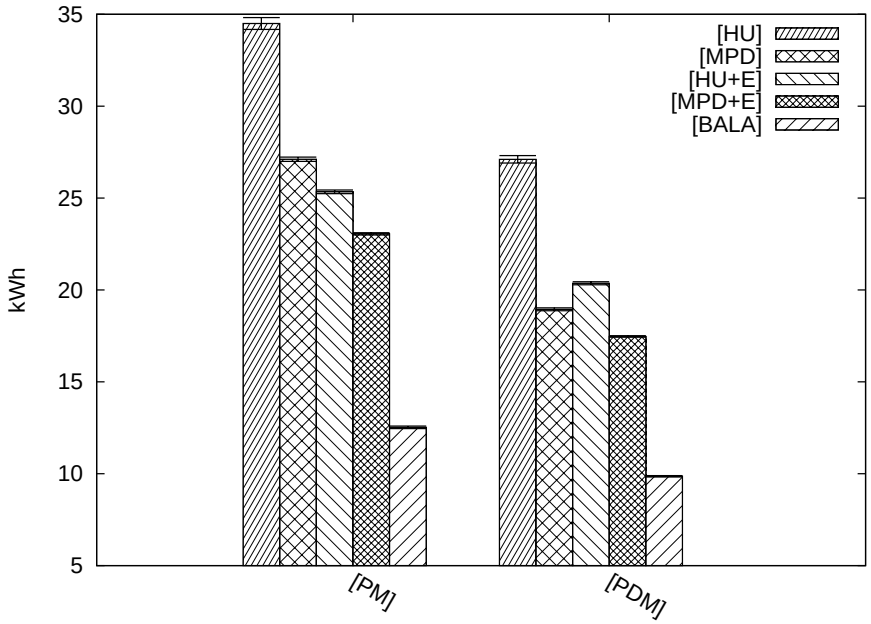

Figura 5.10: Consumo de Energia: *Data Center* Heterogêneo, 100 Servidores

De modo geral, com relação ao `HU`, a adição do `+E` aumentou ligeiramente o *makespan* em *data centers* homogêneos e heterogêneos, e o reduziu ligeiramente em *data centers* mistos; com relação ao `MPD`, o *makespan* foi reduzido em todos os *data centers*, exceto nos $s10$ e $s100$. Comparado a todos os demais, o `BALA` desempenhou melhor em *data centers* não homogêneos.

Nós observamos que devido às migrações mais rápidas, obtidas com o uso de *EBPAs*, existem duas possíveis consequências (A) e (B). Considere dois cenários similares, (i) e (ii),

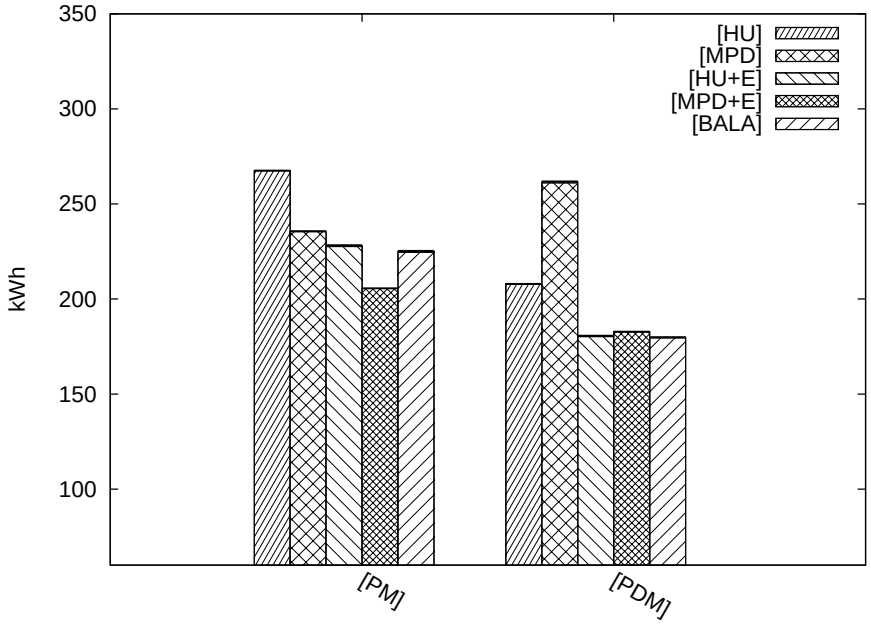

Figura 5.11: Consumo de Energia: *Data Center* Homogêneo, 1.000 Servidores

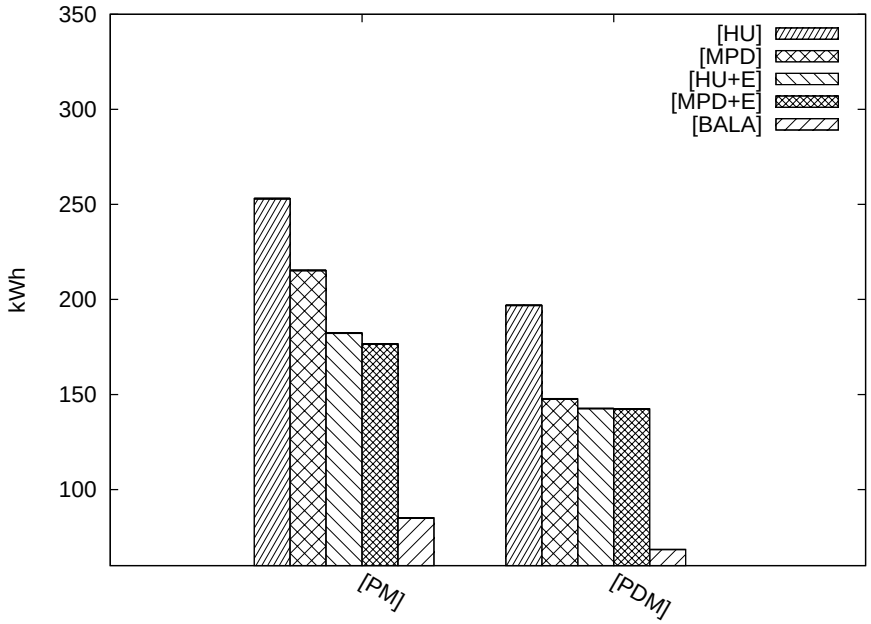

Figura 5.12: Consumo de Energia: *Data Center* Misto, 1.000 Servidores

diferenciados somente pelo *BPA*: (i) usando `SBPA` e (ii) usando um *EBPA*, por exemplo, `+E`.

Caso (A): Se o intervalo de checagem de migrações do *broker* for grande o bastante para suportar pelo menos o tempo consumido pelas migrações despachadas na verificação anterior, e este intervalo for o mesmo em (i) e (ii), então o número de migrações será semelhante entre (i) e (ii). Além disso, devido às migrações mais rápidas, as máquinas virtuais do cenário (ii) apresentarão menor degradação de desempenho durante seus processos de

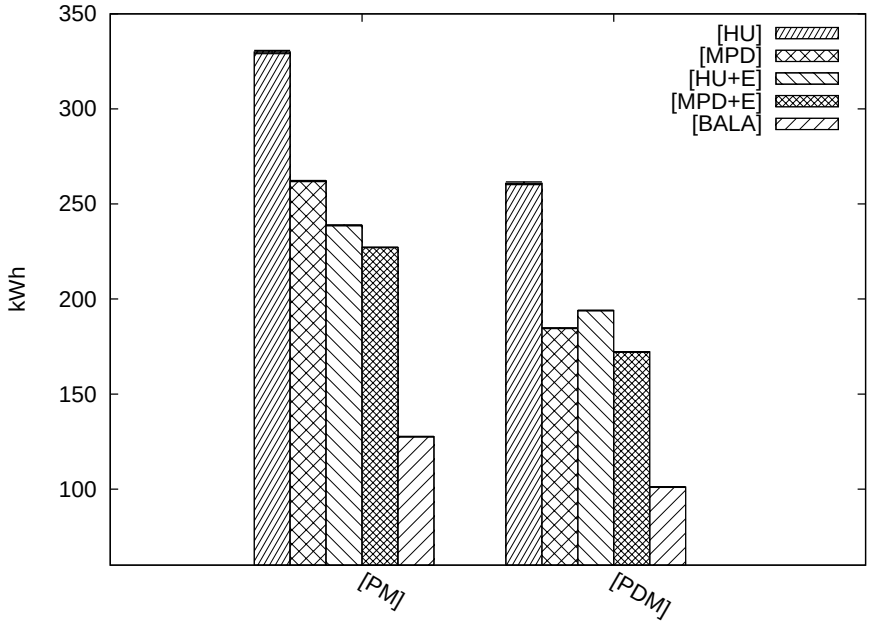

Figura 5.13: Consumo de Energia: *Data Center* Heterogêneo, 1.000 Servidores

migração de máquinas virtuais, acarretando em uma possível redução do *makespan* e do consumo de energia para o cenário (ii) em relação ao cenário (i).

Caso (B): Se o intervalo de verificação de migrações do *broker* ocorrer mais frequentemente do que (A), e.g. se ele for muito curto, ou se o *broker* é notificado pelos servidores no evento de finalização da migração das máquinas virtuais, o número de migrações pode potencialmente aumentar em (ii) quando comparado a (i). Esta maior taxa de migrações pode causar uma redução no consumo de energia, possivelmente em uma maior escala do que no Caso (A), devido ao fato que os servidores, em particular aqueles que são menos eficientes em energia, poderem ser desligados mais cedo. Nos cenários simulados, nós vemos uma predominância de ocorrência deste caso.

Nas Figuras 5.14–5.22, apresentamos o *makespan* para os cenários simulados, constituídos por data homogêneos, mistos e heterogêneos, com tamanhos $s10$, $s100$ e $s1000$.

Em nossa análise, contemplamos comparações sobre consumo de energia e *makespan*, baseados nas inequações apresentadas, dividida em dois estágios. Primeiro, analisamos e comparação os *FHAs* bem conhecidos em suas versões clássicas — `HU` e `MPD` — com suas respectivas versões com o o *EBPA* proposto — `HU+E` e `MPD+E` — com o objetivo de obter dados apurados sobre onde existem ou não melhorias na incorporação de *EBPA*. Posteriormente, com o objetivo de verificar se o método de escalonamento proposto — `BALA` — é uma boa opção para cada cenário, nós comparamos todos estes algoritmos entre si e com o `BALA`. Para fomentar esta análise, apresentamos na Tabela 5.5 dados sobre o consumo de energia para todas as simulações. Na coluna *DC* o tipo do *data center* é listado; o número de servidores está em *Servidores*; *Configuração* remete à configuração/recursos que os algoritmos de escalonamento poderão utilizar; `HU` e `MPD` se referem ao consumo médio de energia dos algoritmos clássicos `HU` e `MPD` nos cenários apresentados em cada linha, bem como *HU CI* e *MPD CI* se referem aos seus respectivos intervalos de confiança em 95%; `HU+E` e `MPD+E` aluz ao consumo médio de energia dos algoritmos `HU` e `MPD` trabalhando em

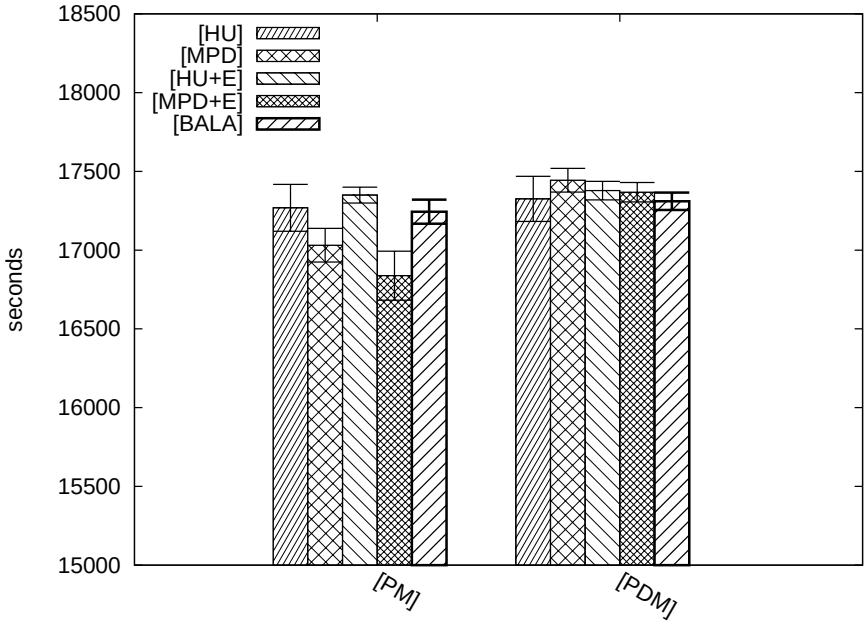

Figura 5.14: *Makespan*: *Data Center* Homogêneo, 10 Servidores

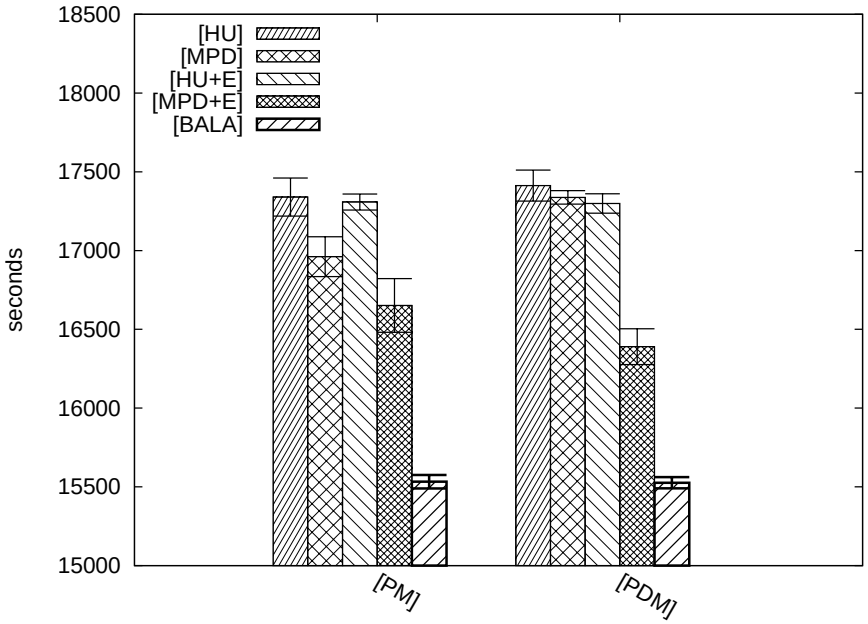

Figura 5.15: *Makespan*: *Data Center* Misto, 10 Servidores

paralelo como o `+E`, bem como *`HU+E` CI* e *`MPD+E` CI* são seus respectivos intervalos de confiança em 95%; *`BALA`* refere ao consumo médio de energia pelo método de escalonamento de máquinas virtuais proposto neste capítulo, e *`BALA` CI* ao seu intervalo de confiança em 95%. Nós também apresentamos na Tabela 5.5 algumas comparações, baseadas na Inequação (3.3), que permite determinar se um algoritmo é melhor do que outro: *CmpHU* se refere à comparação se é *verdade*iro que $\texttt{HU+E} + CI < \texttt{HU} - CI$; em outras palavras, se adicionar o `+E` o torna, para o mesmo conjunto de parâmetros, melhor do que o algoritmo clássico `HU`; bem como *CmpMPD* responde se é *verdadeiro* que $\texttt{MPD+E} + CI < \texttt{MPD} - CI$,

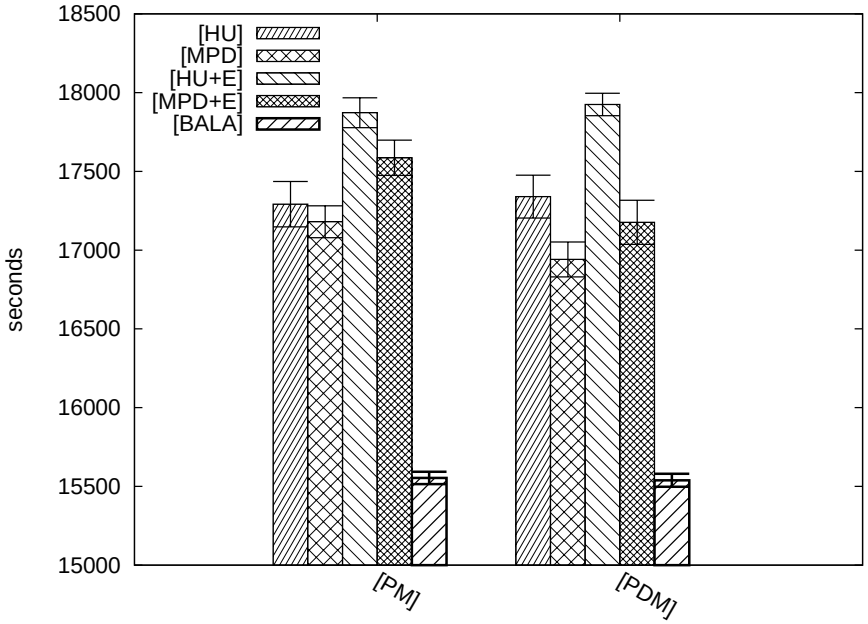

Figura 5.16: *Makespan*: *Data Center* Heterogêneo, 10 Servidores

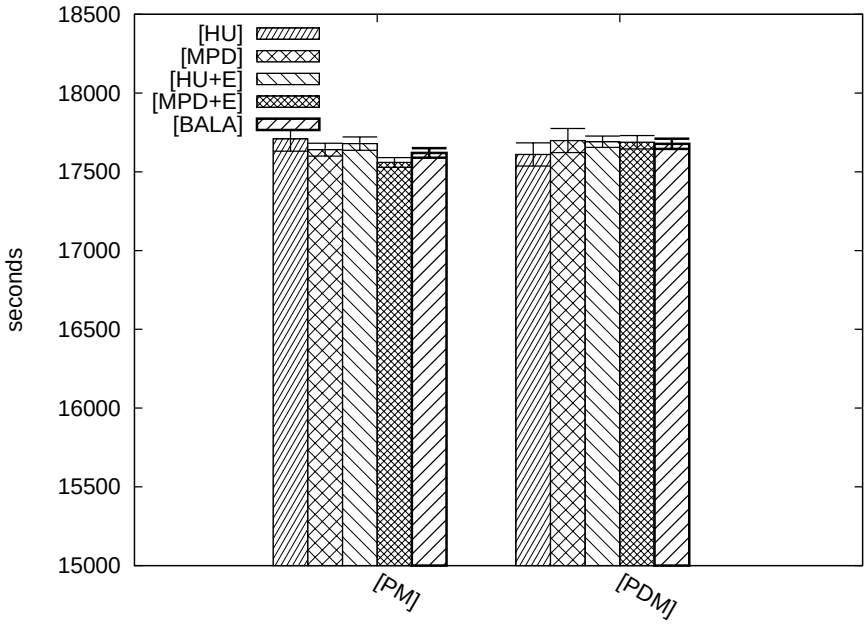

Figura 5.17: *Makespan*: *Data Center* Homogêneo, 100 Servidores

i.e., se adicionar o `+E` o torna, para o mesmo conjunto de parâmetros, melhor do que o algoritmo clássico `MPD`. Na coluna *B.BALA* nós apresentamos se o método apresentado neste capítulo pode ser considerado, em frente aos outros parâmetros estabelecidos, melhor do que todos os demais algoritmos. i.e., se $\texttt{BALA} + CI < MIN(\texttt{ALL} - CI)$, onde `ALL` representa todos os outros algoritmos; e a coluna *NB.BALA* representa o oposto, se algum outro algoritmo é melhor do que o `BALA` para o cenário analisado: a interpretação que pode ser feita destas duas colunas é: se *B.BALA* e *NB.BALA* são ambas *falsas*, então o `BALA` empata com o algoritmo melhor avaliado (dentre `HU`, `MPD`, `HU+E` e `MPD+E`) para o

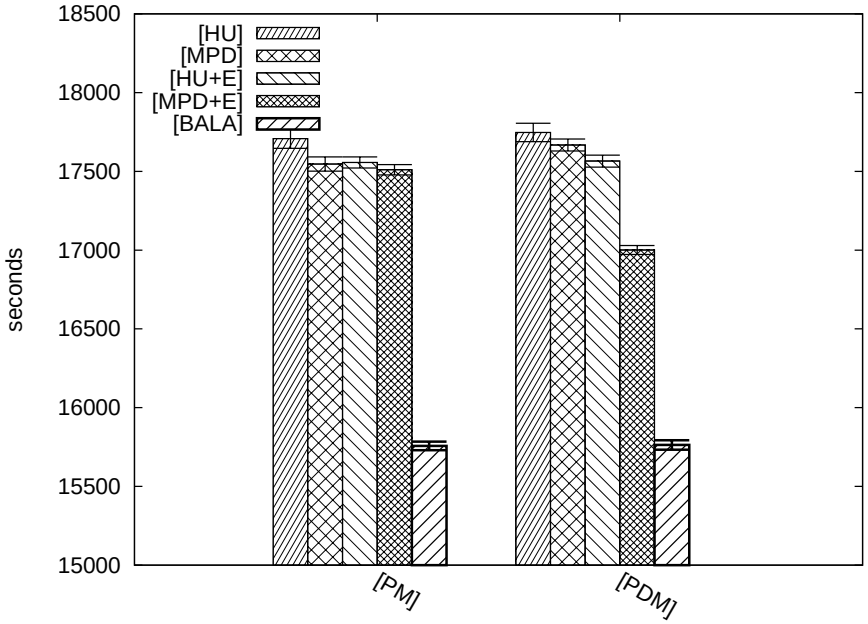

Figura 5.18: *Makespan*: *Data Center* Misto, 100 Servidores

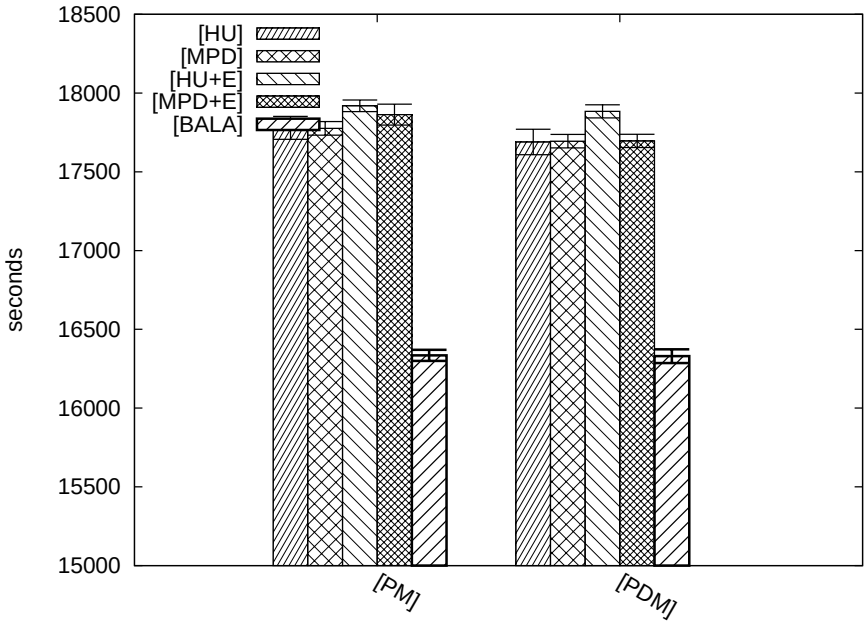

Figura 5.19: *Makespan*: *Data Center* Heterogêneo, 100 Servidores

referido cenário, i.e., se não é possível dizer que um algoritmo específico é melhor do que o outro ou vice-versa; se *NB.BALA* é *verdadeiro* (e portanto *B.BALA* é *falso*) então existe pelo menos um algoritmo competido que é melhor do que o `BALA`; se *B.BALA* é *verdadeiro* (e portanto *NB.BALA* é *falso*) então para aquele cenário o `BALA` é o melhor algoritmo a ser usado. Não é possível para *B.BALA* e *NB.BALA* serem ambos *verdadeiro*s, uma vez que o `BALA` não pode ser melhor do que os demais e vice-versa simultaneamente.

Como podemos observar nas colunas `CmpHU` e `CmpMPD`, a inclusão do `+E` trouxe uma significante eficiência em energia, para todos os cenários estudados, por permitir migrações

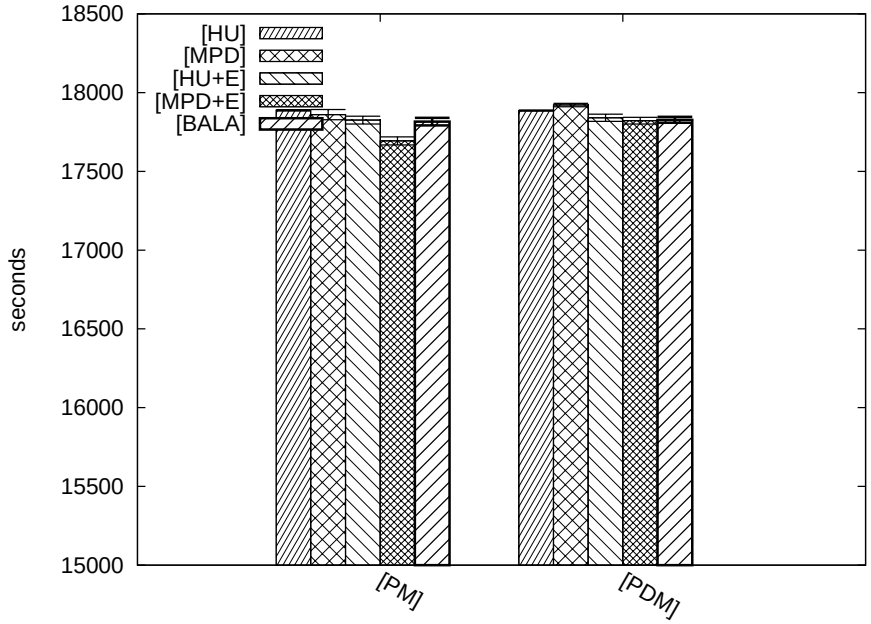

Figura 5.20: *Makespan*: *Data Center* Homogêneo, 1.000 Servidores

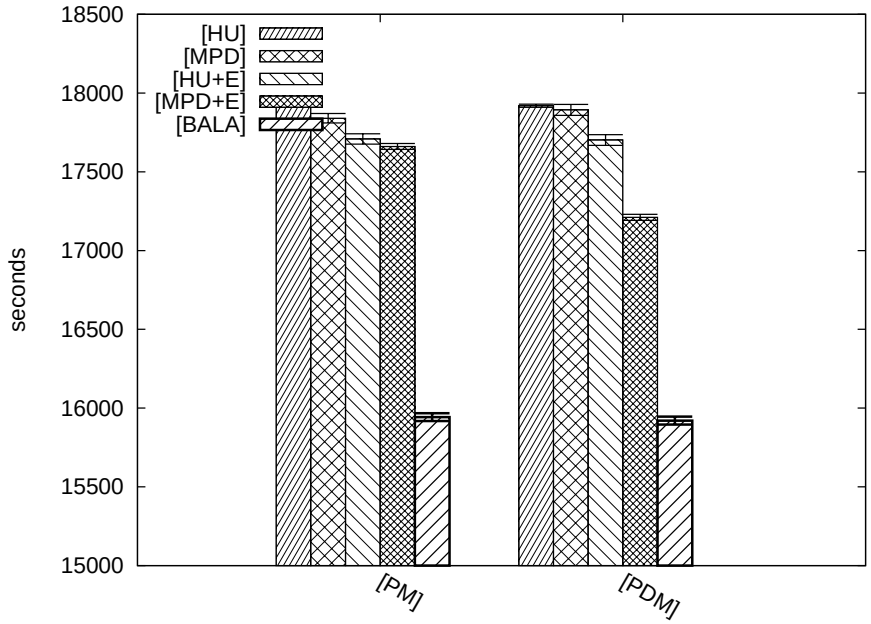

Figura 5.21: *Makespan*: *Data Center* Misto, 1.000 Servidores

mais rápidas de máquinas virtuais. De fato, no caso do `HU`, nós tivemos uma redução no consumo de energia para *data centers* homogêneos, mistos e heterogêneos de, em média, pelo menos 14%, 29% e 26% respectivamente. Para o caso do `MPD`, a redução no consumo de energia para *data centers* homogêneos, mistos e heterogêneos foi, em média, de pelo menos 23%, 11% e 10% respectivamente. Portanto, em média o `+E` reduziu o consumo de energia do `HU` em pelo menos 23%, e de 15% para o `MPD`. Em outras, palavras, a implementação do `+E` para algoritmos de escalonamento de máquinas virtuais trouxe economia de energia para os cenários estudados: para o `HU` variando de 0,54 (= 2,87 −

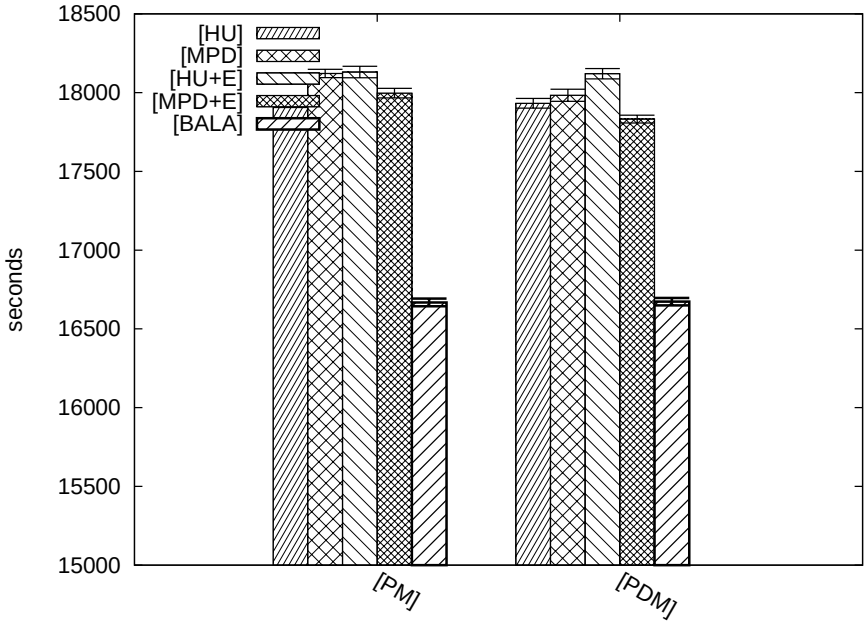

Figura 5.22: *Makespan*: *Data Center* Heterogêneo, 1.000 Servidores

2,33) kWh para *data center* $s10$ homogêneo usando `PDM` até 91,17 ($= 329{,}90 - 238{,}73$) kWh para *data center* $s1000$ heterogêneo com configuração `PM`; e para o `MPD` esta variação foi de 0,13 ($= 1{,}88 - 1{,}75$) kWh para *data center* $s10$ misto com configuração `PDM` até 78,75 ($= 261{,}53 - 182{,}78$) kWh para *data center* homogêneo $s1000$ `PDM`.

Com relação ao *FHA* proposto neste capítulo, `BALA`, quando comparado aos demais, observamos que: ele não é a melhor opção em *data centers* homogêneos sob configuração `PM`, pois ele se comporta pior que o `MPD-PM` em média pelo menos 12%, 9% e 8%, para tamanhos $s10$, $s100$ e $s1000$, respectivamente. No entanto, para os mesmos tipos e tamanhos de *data centers*, sob a configuração `PDM` — a mais eficiente em energia — um empate ocorre. Para todos os outros casos, isto é, *data centers* mistos, heterogêneos, de quaisquer tamanhos — $s10$, $s100$ e $s1000$ — e com qualquer configuração de ciência de energia — `PM` e `PDM` — o `BALA` se mostrou a melhor opção, com uma redução no consumo de energia, em média, de pelo menos 43% quando comparado aos demais.

Com relação ao *makespan*, nós realizamos uma análise baseada nas Inequações (3.4) e (3.5), para verificar como o `+E` impacta os *FHAs*. Nós apresentamos na Tabela 5.6 uma análise comparativa, onde cada linha representa cada cenário, e as colunas têm os seguintes significados: *DC* ser refere tipo de *data center*; *Servidores* indica o número de servidores para o cenário em questão; ***HU*** e ***MPD*** referenciam o *makespan* médio mensurado pelos algoritmos clássicos `HU` e `MPD`, bem como seus intervalos de confiança em 95% são apresentados nas colunas ***HU CI*** e ***MPD CI***; ***HU+E*** e ***MPD+E*** se referem ao *makespan* dos algoritmos `HU` e `MPD`, executados com o *EBPA* `+E`, bem como ***HU+E CI*** e ***MPD+E CI*** indicam seus intervalos de confiança em 95%; ***BALA*** se refere ao método de escalonamento proposto neste capítulo, e ***BALA CI*** indica seu intervalo de confiança em 95%. Nós observamos na coluna *CmpHU* se é *verdade*iro que $HU - CI > 2 \times HU + E - CI$ (Inequação (3.4)) *e* $HU{+}CI > 2{\times}HU{+}E{+}CI$ (Inequação (3.5)), o que, se for *verdadeiro*, pode ser entendido como "a adição do *EBPA* `+E` tornou o `HU+E` melhor do que sua versão clássica (`HU`)"; e

na coluna *CmpMPD* é verificado se as afirmações $MPD - CI > 2 \times MPD + E - CI$ (Inequação (3.4)) *e* $MPD + CI > 2 \times MPD + E + CI$ (Inequação (3.5)) são *verdadeiras*. A coluna *B.BALA* informa se é *verdadeiro* que o `BALA` é o melhor entre os algoritmos apresentados, i.e., se a Inequação (5.13) é satisfeita, onde $MIN(ALL - CI)$ se refere ao menor valor encontrado entre todos os algoritmos avaliados (`HU`, `MPD`, `HU+E` e `MPD+E`) subtraídos de seus respectivos intervalos de confiança em 95%.

$$BALA + CI < MIN(ALL - CI) \tag{5.13}$$

Como nós podemos observar, considerando as comparações de *makespan* apresentadas nas Inequações (3.4) e (3.5)), a adição do *EBPA* `+E` ao `HU` e `MPD` não trouxe mudanças significativas no *makespan*.

Para as simulações executadas, não foram observadas violações de SLA previstas na Seção 5.4.

Comparando o `BALA` com o `HU` e o `MPD`, nós constatamos que no caso de *data centers* homogêneos ele foi, em média, 1% mais lento. No entanto, ele desempenhou 7% mais rápido em *data centers* mistos e heterogêneos e considerando as Inequações definidas para *makespan*, não é possível afirmar que o `BALA` é inaceitavelmente mais lento que os demais algoritmos ou vice-versa.

Em resumo, os resultados sugerem que:

- O emprego de *EBPA* (neste caso, o `+E`) pode prover eficiência em energia, devido a um menor impacto no desempenho pelas migrações;
- Para os cenários estudados, o `BALA-PDM` é arguivelmente uma boa opção para reduzir o consumo de energia para *data centers* homogêneos, especialmente se a configuração `PDM` for aplicada;
- Nos cenários estudados, `BALA-PDM` é a melhor opção para reduzir o consumo de energia para *data centers* mistos e heterogêneos;
- Nos cenários estudados, o `BALA-PDM` é a melhor opção para redução do *makespan* em todos os *data centers* não-homogêneos.

Portanto, tanto o `BALA` quanto o `+E` são opções viáveis em termos de eficiência em energia, enquanto não impactando negativamente o *makespan*.

5.8 Comentários Finais

Neste capítulo apresentamos um método de escalonamento ciente de energia de máquinas virtuais envolvendo dois algoritmos: (i) o *FHA* `BALA`, executado pelo *broker*, para encontrar servidores para máquinas virtuais; e outro, (ii) o *BPA* `+E`, executado pelos servidores, para prover largura de banda adicional para a migração de máquinas virtuais em cenários com SLAs restritivos em largura de manda.

O algoritmo para encontrar servidores para máquinas virtuais é baseado em um algoritmo previamente existente ([72]), e com o conhecimento obtido sobre os os impactos

do aumento das larguras de banda dos *data centers*, nós conseguimos o melhorar com o objetivo de atingir maior eficiência em energia.

Além disso, o *BPA* proposto foi implementado de um modo versátil, de alta granularidade, podendo ser integrado com relativa facilidade a outros algoritmos de escalonamento de máquinas virtuais.

O método de alocação de máquinas virtuais e provisionamento de largura de banda é implantável em qualquer *data center* operando em nuvem, com resultados de eficiência em energia similar a outros métodos eficientes em energia para *data centers* homogêneos, mas com ganhos notáveis para *data centers* não homogêneos, sem impacto significativo no *makespan*. Os ganhos obtidos com este método, quando comparado aos outros estudados, foi capaz de reduzir o consumo de energia em 43% para *data centers* não-homogêneos.

Portanto, a ciência de largura de banda, em *data centers* em servidores que operam sob distintas taxas de transmissão, pode ser usada com o objetivo de reduzir o consumo de energia, pois ela reduz os tempos de migrações de máquinas virtuais. Estes tempos reduzidos tornam possível evitar sub-processamento devido à finalização acelerada e o consequente desligamento adiantado de servidores. A soma destes aspectos contribui substancialmente para maior eficiência em energia, e estes podem ser explorados como nós fizemos nos métodos de escalonamento.

Tabela 5.5: Análise do Consumo de Energia — Valores Numéricos em kWh com CI apresentado usando ±

DC	**# Servs.**	**Conf.**	HU	HU+E	MPD	MPD+E	BALA	**Cmp**HU	**Cmp**MPD	**B,**BALA	**NB,**BALA
Homogêneo	10	PM	3,82 ± 0,12	3,03 ± 0,02	3,45 ± 0,07	2,47 ± 0,13	2,97 ± 0,03	Verdade	Verdade	Falso	Verdade
		PDM	2,87 ± 0,07	2,33 ± 0,01	3,32 ± 0,06	2,31 ± 0,01	2,30 ± 0,01	Verdade	Verdade	Falso	Falso
	100	PM	27,75 ± 0,10	24,05 ± 0,11	24,31 ± 0,08	21,28 ± 0,07	23,67 ± 0,12	Verdade	Verdade	Falso	Verdade
		PDM	21,46 ± 0,08	18,93 ± 0,08	26,86 ± 0,19	18,96 ± 0,08	18,99 ± 0,06	Verdade	Verdade	Falso	Falso
	1.000	PM	267,53 ± 0,21	228,04 ± 0,41	235,57 ± 0,24	205,61 ± 0,15	225,00 ± 0,48	Verdade	Verdade	Falso	Verdade
		PDM	207,95 ± 0,15	180,53 ± 0,35	261,53 ± 0,47	182,78 ± 0,18	179,78 ± 0,28	Verdade	Verdade	Verdade	Falso
Misto	10	PM	3,72 ± 0,12	2,44 ± 0,01	2,80 ± 0,09	1,98 ± 0,11	1,33 ± 0,03	Verdade	Verdade	Verdade	Falso
		PDM	2,79 ± 0,07	1,87 ± 0,01	1,88 ± 0,03	1,75 ± 0,09	1,03 ± 0,02	Verdade	Verdade	Verdade	Falso
	100	PM	26,71 ± 0,10	18,81 ± 0,06	22,03 ± 0,12	18,19 ± 0,05	9,15 ± 0,11	Verdade	Verdade	Verdade	Falso
		PDM	20,58 ± 0,09	14,65 ± 0,06	15,44 ± 0,08	14,50 ± 0,07	7,33 ± 0,08	Verdade	Verdade	Verdade	Falso
	1.000	PM	253,04 ± 0,32	182,30 ± 0,19	215,30 ± 0,27	176,61 ± 0,11	85,04 ± 0,08	Verdade	Verdade	Verdade	Falso
		PDM	196,93 ± 0,24	142,64 ± 0,15	147,66 ± 0,18	142,36 ± 0,13	68,47 ± 0,04	Verdade	Verdade	Verdade	Falso
Heterogêneo	10	PM	4,45 ± 0,13	3,43 ± 0,02	3,73 ± 0,06	3,14 ± 0,08	1,50 ± 0,03	Verdade	Verdade	Verdade	Falso
		PDM	3,35 ± 0,11	2,73 ± 0,01	2,15 ± 0,05	2,27 ± 0,09	1,14 ± 0,02	Verdade	Verdade	Verdade	Falso
	100	PM	34,50 ± 0,32	25,34 ± 0,10	27,11 ± 0,11	23,05 ± 0,06	12,52 ± 0,07	Verdade	Verdade	Verdade	Falso
		PDM	27,11 ± 0,20	20,36 ± 0,08	18,95 ± 0,08	17,47 ± 0,04	9,86 ± 0,04	Verdade	Verdade	Verdade	Falso
	1.000	PM	329,90 ± 0,87	238,73 ± 0,26	262,04 ± 0,28	227,14 ± 0,14	127,59 ± 0,17	Verdade	Verdade	Verdade	Falso
		PDM	260,92 ± 0,73	193,94 ± 0,17	184,64 ± 0,19	172,20 ± 0,12	101,04 ± 0,12	Verdade	Verdade	Verdade	Falso

Tabela 5.6: Análise de *Makespan* — Valores numéricos em segundos com intervalo de confiança reportado como ±

								Cmp		
DC	**# Servs.**	**Cnf.**	HU	HU+E	MPD	MPD+E	BALA	HU	MPD	**B,**BALA
Homogêneo	10	PM	17.268,44 ± 148,77	17.349,33 ± 50,19	17.031,06 ± 107,02	16.837,78 ± 155,72	17.244,11 ± 76,57	Falso	Falso	Falso
		PDM	17.325,28 ± 143,20	17.377,88 ± 58,71	17.443,65 ± 75,07	17.367,60 ± 61,54	17.309,85 ± 55,29	Falso	Falso	Falso
	100	PM	17.709,37 ± 78,42	17.678,87 ± 42,16	17.640,37 ± 40,73	17.559,46 ± 31,08	17.620,05 ± 30,51	Falso	Falso	Falso
		PDM	17.610,03 ± 73,45	17.691,03 ± 35,66	17.698,53 ± 76,15	17.688,03 ± 42,28	17.677,80 ± 32,39	Falso	Falso	Falso
	1.000	PM	17.887,53 ± 4,45	17.825,20 ± 24,66	17.860,03 ± 32,29	17.693,37 ± 25,67	17.815,96 ± 24,27	Falso	Falso	Falso
		PDM	17.885,87 ± 3,76	17.840,20 ± 23,08	17.921,37 ± 10,81	17.821,70 ± 21,21	17.827,30 ± 19,04	Falso	Falso	Falso
Misto	10	PM	17.339,94 ± 120,76	17.308,21 ± 50,60	16.961,14 ± 126,43	16.651,82 ± 170,40	15.532,75 ± 42,94	Falso	Falso	Verdade
		PDM	17.412,76 ± 98,48	17.299,16 ± 61,50	17.337,70 ± 42,52	16.389,94 ± 113,68	15.526,00 ± 35,73	Falso	Falso	Verdade
	100	PM	17.707,53 ± 61,19	17.556,53 ± 35,25	17.546,37 ± 45,08	17.510,53 ± 32,77	15.756,67 ± 27,16	Falso	Falso	Verdade
		PDM	17.746,87 ± 58,37	17.565,03 ± 38,27	17.667,03 ± 38,01	17.000,83 ± 28,38	15.763,08 ± 30,28	Falso	Falso	Verdade
	1.000	PM	17.927,20 ± 11,43	17.708,70 ± 32,62	17.840,03 ± 29,95	17.660,87 ± 18,93	15.942,50 ± 24,92	Falso	Falso	Verdade
		PDM	17.919,20 ± 10,07	17.701,87 ± 34,20	17.893,37 ± 34,77	17.210,80 ± 19,48	15.920,50 ± 24,92	Falso	Falso	Verdade
Heterogêneo	10	PM	17.291,73 ± 144,35	17.872,33 ± 94,55	17.180,36 ± 101,11	17.586,19 ± 112,03	15.553,83 ± 39,23	Falso	Falso	Verdade
		PDM	17.339,83 ± 136,05	17.924,28 ± 71,23	16.941,14 ± 110,83	17.176,60 ± 140,13	15.538,83 ± 41,27	Falso	Falso	Verdade
	100	PM	17.779,20 ± 73,03	17.919,03 ± 36,61	17.775,87 ± 43,30	17.863,20 ± 66,30	16.334,52 ± 35,13	Falso	Falso	Verdade
		PDM	17.689,48 ± 80,89	17.883,70 ± 41,77	17.694,37 ± 42,68	17.696,03 ± 42,02	16.329,56 ± 43,48	Falso	Falso	Verdade
	1.000	PM	17.909,03 ± 3,43	18.130,70 ± 36,30	18.121,70 ± 26,57	17.996,70 ± 30,06	16.667,71 ± 24,30	Falso	Falso	Verdade
		PDM	17.932,20 ± 30,82	18.119,87 ± 32,70	17.982,87 ± 38,40	17.831,87 ± 24,44	16.672,96 ± 23,61	Falso	Falso	Verdade

Capítulo 6

Topology Energy-Aware Allocator

Vimos até aqui os impactos das redes de alta velocidade no processo de escalonamento de máquinas virtuais. Focamos especialmente na energia, mas analisamos, também, fatores importantes, como *makespan*, tempos de execução de cargas de trabalho e número de migrações. Propomos, ainda, um algoritmo para escalonamento ciente de largura de banda, com ênfase na minimização do consumo de energia, e obtivemos bons resultados, sobretudo em *data centers* heterogêneos.

No entanto, observamos que o aumento da velocidade das redes impacta não somente a largura de banda dos servidores, mas em muito a atuação da topologia da rede em si. Objetivando levar o escalonamento de máquinas virtuais a um novo patamar em economia de energia, decidimos considerar também este nível impactado pelas redes de alta velocidade.

Com base nos trabalhos estudados, e com a adição de uma gama de novas estratégias voltadas aos elementos de núcleo da rede, expandimos a inteligência do processo de escalonamento para tentar minimizar o consumo de energia desligando não somente servidores, mas — com conhecimento da topologia — também partes da rede.

Neste capítulo apresentamos e avaliamos o *Topology Energy-Aware Allocator* (`TEA`), um algoritmo para escalonamento de máquinas virtuais ciente de energia e da topologia da rede, para a computação em nuvem, que objetiva minimizar o consumo de energia em *data centers*, e se possível também o *makespan* e o tempo de execução de cargas individuais.

Este algoritmo tem como escopo atuar em *data centers* de tamanhos arbitrários, homogêneos ou heterogêneos, geodistribuídos ou não, com topologia de rede conhecida. Para atingir o objetivo de minimizar o consumo de energia nestes *data centers*, ele lida com recursos como a gerenciamento distribuído de energia — incluindo o desligamento de servidores ociosos e de comutadores de pacotes — e DVFS.

Para atingir o objetivo de minimizar o consumo de energia, este algoritmo, além de alguns dos recursos clássicos comumente incorporados em algoritmos de escalonamento cientes de energia, utiliza o pré-processamento de ordem de máquinas virtuais: a cada verificação do *broker*, as máquinas virtuais passam por uma ordenação de modo a escalonar, perante critérios pré-definidos, prioritariamente algumas máquinas virtuais em detrimento de outras. Além disso, este algoritmo pode agir reconhecendo a prioridade solicitada pelos usuários objetivando, além de prover qualidade de serviço dando preferência às cargas de prioridades mais elevadas, usar esse recurso para prover economia de energia, de modo a

evitar escalonar máquinas virtuais que não tenham prioridades elevadas em um percentual de servidores com baixa eficiência em energia. Quanto à ciência de topologia, dizemos que este algoritmo analisa aspectos da topologia, em especial as rotas entre os sistemas finais comunicantes dentro dos *data centers* que compõem a nuvem e suas respectivas eficiências em energia, para a determinação de servidores para hospedarem máquina virtuais.

O `TEA` também é projetado para utilizar — ou não, a depender do usuário da nuvem — um estimador de tempo para finalização das máquinas virtuais, objetivando tentar prever e evitar a migração de máquinas virtuais que potencialmente deverão ser finalizadas logo. Um estimador ideal é aquele que seria capaz de determinar, com precisão absoluta, qual é o tempo real que uma máquina virtual consumirá para terminar de processar suas cargas e serem desalocadas. Isso é uma tarefa extremamente difícil — senão impossível. Uma abordagem comum para esta finalidade verificada em *clusters* é solicitar aos próprios usuários do sistema esta estimativa. Esta abordagem pode ser ruim, por várias razões: (i) o usuário pode não ter uma ideia razoável deste tempo, (ii) aumento da burocracia para uso do *data center*, (iii) para o usuário fazer esta estimativa pode-se consumir um tempo precioso. Nossa implementação do `TEA` é realizada usando um modelo estatístico automatizado para cálculo do tempo restante para finalização de uma dada máquina virtual. As principais e visíveis vantagens deste modelo são a agilidade obtida pela eliminação da burocracia do *data center* e a redução do erro humano inerente.

6.1 `TEA` — Funções Auxiliares

Como vimos, a arquitetura do `TEA` atua em duas principais frentes, que são a ordenação do conjunto das máquinas virtuais com as quais o *broker* lida e a determinação de máquinas físicas para hospedarem tais máquinas virtuais. Os algoritmos executados em ambas as frentes utilizam funções auxiliares, as quais tratamos nesta seção.

A função `getRoute(`*source*`, `*destination*`)` retorna uma rota, isto é, um conjunto de interfaces de rede pelas quais pacotes são recebidos no tráfego de dados de um sistema final de origem *source* para um sistema final de destino *destination*. Note que, por essa definição, considerando dois sistemas finais distintos $h0$ e $h1$, temos que `getRoute(`$h0$`, `$h1$`)` $\neq$ `getRoute(`$h1$`, `$h0$`)`. A implementação desta função é simples e dependente de estar registrado no *broker* a topologia da rede. Esta definição que apresentamos é voltada para redes *full-duplex*, mas é possível realizar adaptações para *half-duplex*. Apresentamos no Algoritmo 3 um exemplo código-fonte que pode ser executado por esta função.

Neste algoritmo, o método `getConnectable()` retorna a interface de rede adjacente ao objeto que a executa; o método `getOwner()` retorna o comutador de pacotes ao qual o objeto que representa a interface de rede pertence; e o método `getPortExitTo(`*destination*`)` observa a tabela de encaminhamento/roteamento do comutador de pacotes em questão e retorna a porta de saída para *destination*. Note que esta função é usada um número grande de vezes pelos algoritmos apresentados nas próximas seções, então, por questões de escalabilidade, é recomendado que ele seja implementado com uso de *cache* histórico, de forma a armazenar as rotas entre *source* e *destination* em um mapa de acesso rápido para futuras consultas. Caso estejam sendo executados protocolos que modificam as rotas

Algoritmo 3 TEA: getRoute(*source*, *destination*)

```
1:  function GETROUTE(source, destination)
2:      route ← ∅
3:      currentConnectable ← source.getConnectable()
4:      while currentConnectable ≠ destination do
5:          route.add(currentConnectable)
6:          currentConnectable ← ((Port) currentConnectable).getOwner()
7:                                 .getPortExitTo(destination).getConnectable()
8:      end while
9:      route.add(destination)
10:     return route
11: end function
```

Algoritmo 4 TEA: calculateRouteMaxBw(*source*, *destination*)

```
1:  function CALCULATEROUTEMAXBW(source, destination)
2:      bandwidth ← ∞
3:      for all connectable ∈ getRoute(source, destination) do
4:          link_bw ← min{connectable_bw_down, connectable.getConnectable()._bw_up}
5:          bandwidth ← min{bandwidth, link_bw}
6:      end for
7:      return bandwidth
8:  end function
```

entre *source* e *destination* — como STPs que alteram as rotas dinamicamente — deve-se tornar o cuidado de zerar este *cache* para manutenção de coerência.

A seguir apresentamos o Algoritmo 4, utilizado pelo `TEA` para obtenção da taxa de transmissão máxima entre os sistemas finais de origem *source* e de destino *destination*. Esta largura de banda máxima é igual à menor taxa de transmissão dos enlaces *connectable*–*connectable*`.getConnectable()` — interface de rede – interface de rede adjacente — que compõem uma rota.

Como veremos nas próximas seções, este algoritmo é usado por diversos outros algoritmos para possibilitar o cálculo da eficiência em energia da rede entre um sistema final de origem e um sistema final de destino.

Apresentamos a seguir o Algoritmo 5, que tem por finalidade estimar o tempo médio restante para finalizar uma dada máquina virtual, em segundos.

O algoritmo apresentado inicia fazendo a simples verificação se, de fato, o estimador está configurado como ativado (Linha 2). Caso não esteja, é retornado um valor infinito positivo que, para efeitos práticos, significa que o estimador não está funcionando.

Na sequência (Linha 5), é verificado se a máquina virtual em questão está processando ou recebendo alguma carga. Caso não esteja — e.g. a máquina virtual já tenha terminado de processar todas as cargas à ela designada — é retornado um valor 0.

A seguir (Linha 8), definimos a variável *vmMips*, que representa o MIPS de uma máquina virtual. Existem duas situações aqui: caso (A) a máquina esteja alocada em um servidor, essa variável receberá o MIPS efetivo da máquina virtual em questão, ou seja, a quantidade de MIPS que o servidor hospedeiro de fato entrega; ou caso (B) a

Algoritmo 5 TEA: estimateAverageTimeRemainingToFinish(vm)

```
1:  function estimateAverageTimeRemainingToFinish(vm)
2:      if ¬ finishTimeEstimatorEnabled then
3:          return ∞
4:      end if
5:      if vm.getWorkloads().isEmpty() then
6:          return 0.0
7:      end if
8:      if vm.getState() = OPERATIONAL then
9:          vmMips ← vm.getMipsEffective()
10:     else
11:         vmMips ← vm.getMipsTotal()
12:     end if
13:     return getHistAvgMiPerWl(userId) × vm.getWorkloads()_total / vmMips
14: end function
```

máquina esteja *offline* — e.g. ainda não tenha sido submetida para processamento — nessa situação, a variável vai receber o valor de MIPS requisitado pela máquina virtual em questão.

Uma vez definida essa variável, finalmente vamos ao passo final (Linha 13). Neste passo, a função `getHistAvgMiPerWl(`$userId$`)` retorna a média histórica referente ao identificador do usuário $userId$ responsável pela solicitação de vm. Caso o *data center* não esteja programado para identificar usuários, ou que o usuário identificado por $userId$ não tenha processado nenhuma carga no *data center*, então esta função poderá retornar os dados históricos de todos os usuários. O parâmetro $vm.\texttt{getWorkloads()}_{total}$ retorna o número de cargas de trabalho que vm está processando em paralelo.

O `TEA` trabalha com um conceito mais amplo de eficiência em energia do que os algoritmos anteriores. O conceito clássico da eficiência em energia é dado em função da capacidade de processamento de um servidor, conforme a Equação (5.4). O `TEA` pode considerar a eficiência em energia $EnergyEfficiency$ em função de recursos computacionais $Resource$ e da potência consumida $Power$, conforme a Equação (6.1).

$$EnergyEfficiency = \frac{Resource}{Power} \tag{6.1}$$

Dentre os recursos computacionais que podem ser trabalhados pelo `TEA` estão: MIPS, RAM e largura de banda. As funções explícitas que realizam cálculos de eficiência em energia — `calculateEnergyEfficiencyMips(`$host$`)`, `calculateEnergyEfficiencyRam(`$host$`)` e `calculateEnergyEfficiencyBw(`$host$`)` — calculam a eficiência em energia conforme a Equação (6.1).

Além dos recursos computacionais básicos supramencionados, o `TEA` também realiza o cálculo da eficiência em energia consolidada da rede para uma rota entre dois sistemas finais *source* e *destination*. Este cálculo é feito conforme o Algoritmo 6.

Sumarizadamente, o Algoritmo 6 trata, portanto, a eficiência em energia de uma dada rota como sendo proporcional à largura de banda que ela oferece e inversamente

Algoritmo 6 TEA: calculateEnergyEfficiencyNetwork(*source*, *destination*)

```
1: function CALCULATEENERGYEFFICIENCYNETWORK(source, destination)
2:     route ← getRoute(source, destination)
3:     powerSum ← 0
4:     for all connectable ∈ route do
5:         switch ← connectable.getOwner()
6:         powerSum ← powerSum + switch_max_power
7:     end for
8:     return calculateRouteMaxBw(source, destination) / powerSum
9: end function
```

Algoritmo 7 TEA: estPwAfterAlloc(*host*, *vm*)

```
1: function ESTPWAFTERALLOC(host, vm)
Require: vm ∉ host
2:     mipsAfterAlloc ← host_mips_requested_by_vms + vm.getMipsTotal()
3:     usageAfterAlloc ← min{1.0, mipsAfterAlloc/host.getMipsTotal()}
4:     return host.getPowerModel().getPower(usageAfterAlloc);
5: end function
```

proporcional ao consumo consolidado de potência dos comutadores de pacotes *powerSum* pelos quais ela passa.

O `TEA` também tenta prever a potência consumida por um servidor durante o processo de escalonamento. Esta previsão pode ser feita de várias maneiras e, dentre elas, consideramos razoável ter um modelo de potência para o servidor em questão. Apresentamos o Algoritmo 7 que tem por objetivo estimar a potência consumida por um servidor *host* após a alocação de uma máquina virtual *vm*.

Sabe-se que o principal elemento de potência consumida por um servidor usualmente é o processador. O algoritmo apresentado visa estimar o consumo de potência de um servidor em função dos MIPS requisitados por uma dada máquina virtual. Inicialmente, é verificado o total de MIPS *mipsAfterAlloc* que será requisitado do servidor após a alocação de *vm*, o que pode ser calculado pela soma do total de MIPS requisitado por todas as máquinas virtuais que este servidor hospeda com a quantidade de MIPS requisitada por *vm*. Note que este valor de requisição pode ser maior que o valor total de MIPS do servidor — o que neste caso acarreta em uma perda de desempenho das máquinas virtuais pelo fato do servidor não conseguir prover todos os MIPS solicitados. Na sequência, é calculada a utilização percentual *usageAfterAlloc* que o servidor terá após a alocação de *vm*. Por fim, é estimado utilizando um modelo de potência qual é o consumo de potência que o servidor ficará após a alocação da máquina virtual. Considerando um modelo de potência com DVFS, por exemplo, pode-se estimar este consumo com a Equação (5.3).

De forma análoga à análise da potência consumida após a alocação de uma dada máquina virtual, podemos também estimar a potência consumida após a desalocação de uma dada máquina virtual conforme mostramos no Algoritmo 8.

Inicialmente, é verificado o total de MIPS *mipsEstAfterDealloc* que será requisitado do servidor após a desalocação de *vm*, que pode ser calculado pela subtração do total de MIPS requisitado por todas as máquinas virtuais que este servidor hospeda com a

Algoritmo 8 TEA: estimatePowerAfterDeallocation(*host*, *vm*)

1: **function** ESTIMATEPOWERAFTERDEALLOCATION(*host*, *vm*)
Require:
vm ∈ *host*
2: $mipsEstAfterDealloc \leftarrow \min\{host$.`getMipsTotal()`,
3: $host_{mips\_requested\_by\_vms} - vm$.`getMipsTotal()`$\}$
4: $utilizationAfterDealloc \leftarrow mipsEstAfterDealloc/host$.`getMipsTotal()`
5: **return** *host*.`getPowerModel().getPower(`$utilizationAfterDealloc$`)`
6: **end function**

quantidade de MIPS requisitada por *vm*. Na sequência, é calculada a utilização percentual $usageAfterAlloc$ que o servidor ficará após a alocação de *vm*. Por fim, é estimado utilizando um modelo de potência qual é o consumo de potência que o servidor ficará após a alocação da máquina virtual. Considerando um modelo de potência com DVFS, por exemplo, pode-se estimar este consumo com a Equação (5.3), com uma consideração adicional: caso o servidor fique com uma utilização de 0% — e.g. não hospedar nenhuma máquina virtual — ele poderá ser desligado — ou seja, o consumo será de 0 W após o desligamento considerando o emprego de `P` no processo de escalonamento.

6.2 `TEA` — Pré-processamento de Máquinas Virtuais

Quando um *broker* está trabalhando com um conjunto de máquinas virtuais, em processamento ou não, o `TEA` atua inicialmente com o intuito de não trabalhar cegamente com a abordagem FIFO. Diferentemente de outros algoritmos, que quando realizam somente uma ordenação — i.e. se realizam — com base na prioridade das máquinas virtuais, o `TEA` faz um pré-processamento completo do conjunto de máquinas virtuais para definir a ordem com a qual ele lidará. Apresentamos no Algoritmo 9 o algoritmo de comparação usado para a ordenação deste conjunto. Esse algoritmo recebe como parâmetro duas máquinas virtuais a serem comparadas, e retorna um valor negativo caso a primeira máquina virtual deve ser ordenada com prioridade, um valor positivo caso a segunda máquina virtual deva ter prioridade, ou 0 caso a prioridade de ambas deva ser a mesma.

O algoritmo de pré-processamento inicia verificando se as máquinas virtuais estão *online* ou *offline* (Linha 2). A prioridade nessa situação é dada para máquinas que estejam *offline*, de modo a permitir um despacho acelerado delas — maior responsividade do *data center*. Esta é uma decisão que na verdade vai contra os princípios da eficiência em energia, pois tende a ligar um maior número de servidores para hospedar as máquinas virtuais. Por outro lado, este comportamento faz com que as requisições das máquinas virtuais sejam atendidas mais rapidamente e, assim, consideramos que este *trade-off* é desejável — pois tende a reduzir substancialmente o *makespan*. Caso não haja uma saída neste desvio condicional, então existe a implicação que deste ponto para frente no código o estado de ambas as máquinas virtuais é o mesmo, ou seja, ambas estão *offline* ou ambas estão *online*.

Na sequência (Linha 10), é checada a prioridade das máquinas virtuais *offline*, e aquela que possuir prioridade mais alta receberá a preferência. Isso garante uma submissão

Algoritmo 9 TEA: compare(vm0, vm1)

```
1:  function COMPARE(vm0, vm1)
2:      if vm0.getState() ≠ vm1.getState() then
3:          if vm0.getState() = INOPERABLE_OFFLINE then
4:              return -1
5:          else                          ▷ if vm1.getState() = INOPERABLE_OFFLINE
6:              return 1
7:          end if
8:      end if
9:      if (vm0.getState() = INOPERABLE_OFFLINE)
10:                              & (vm0.getPriority() ≠ vm1.getPriority()) then
11:         if vm0.getPriority() = HIGH then
12:             return -1
13:         end if
14:         if vm1.getPriority() = HIGH then
15:             return 1
16:         end if
17:         if vm0.getPriority() = NORMAL then
18:             return -1
19:         end if
20:         if vm1.getPriority() = NORMAL then
21:             return 1
22:         end if
23:     end if
```

acelerada das máquinas virtuais com maior prioridade definida pelo usuário. Esta também não é uma decisão baseada na economia de energia, mas em atendimento rápido aos anseios dos usuários da nuvem, no sentido de minimizar o tempo de processamento de máquinas virtuais com alta prioridade. Note que como assumimos três prioridades de máquinas virtuais (alta, média e baixa), não precisamos fazer a comparação dentro do aninhamento para verificar se a prioridade da máquina virtual é baixa, pois esta é uma condição impossível — duas máquinas virtuais com diferentes prioridades e uma delas não ser alta ou normal após passar por todos os critérios de desempate neste nível de aninhamento.

Após isso, é verificado, na Linha 25, a prioridade das máquinas virtuais *online* e, ao contrário da situação anterior, aquela que possuir prioridade mais baixa receberá a preferência. Essa decisão é tomada com o objetivo de migrar preferencialmente máquinas virtuais de prioridades mais baixas para evitar o impacto de desempenho nas máquinas virtuais de prioridades mais altas. Mais uma vez, esta decisão não é baseada na economia de energia, mas em atendimento rápido aos anseios dos usuários da nuvem, no sentido também de minimizar o tempo de processamento de máquinas virtuais com alta prioridade.

Caso a execução passe deste ponto, existem duas garantias: ambas as máquinas virtuais que estão sendo comparadas apresentam (i) o mesmo estado e (ii) a mesma prioridade.

```
24:  if (vm0.getState() = OPERATIONAL)
25:                          & (vm0.getPriority() ≠ vm1.getPriority()) then
26:      if vm0.getPriority() = HIGH then
27:          return 1
28:      end if
29:      if vm1.getPriority() = HIGH then
30:          return -1
31:      end if
32:      if vm0.getPriority() = NORMAL then
33:          return 1
34:      end if
35:      if vm1.getPriority() = NORMAL then
36:          return -1
37:      end if
38:  end if
```

```
39:  if vm0.getState() = OPERATIONAL then
40:      h0 ← vm0.getHostProcessing()
41:      h1 ← vm1.getHostProcessing()
42:      h0EE ← calculateEnergyEfficiencyMips(h0)
43:      h1EE ← calculateEnergyEfficiencyMips(h1)
44:      if h0EE > h1EE then
45:          return 1
46:      else if h0EE < h1EE then
47:          return -1
48:      end if
49:      h0EE ← calculateEnergyEfficiencyRam(h0)
50:      h1EE ← calculateEnergyEfficiencyRam(h1)
51:      if h0EE > h1EE then
52:          return 1
53:      else if h0EE < h1EE then
54:          return -1
55:      end if
```

O algoritmo começa a, de fato, se preocupar com a eficiência a partir da Linha 39. Neste trecho, ele verifica se, caso as máquinas virtuais estejam operacionais, ou seja, hospedadas e ativas, qual o servidor é o menos eficiente em energia. Primeiramente, é verificado entre os servidores que hospedam $vm0$ e $vm1$, respectivamente $h0$ e $h1$, se algum deles é mais eficiente energeticamente que o outro. Aqui o algoritmo primeiro compara a eficiência em energia calculada tomando como base o MIPS dos servidores (Linha 42), de acordo com a Equação (5.4) e, caso ocorra um empate, usa-se a Equação (6.2) — onde $host_{ram}$ representa a quantidade de RAM que um servidor possui — para comparar a eficiência em energia $EnergyEfficiency$ proporcional à quantidade de RAM dos servidores em relação à potência consumida por estes $host_{max\_pow}$ (Linha 49).

$$EnergyEfficiency = \frac{host_{ram}}{host_{max\_pow}} \tag{6.2}$$

```
56:     if estimateAverageTimeRemainingToFinish(vm0)
57:                     < estimateAverageTimeRemainingToFinish(vm1) then
58:         return 1
59:     end if
60:     if estimateAverageTimeRemainingToFinish(vm0)
61:                     > estimateAverageTimeRemainingToFinish(vm1) then
62:         return -1
63:     end if
64: end if
65: if vm0.getState() = INOPERABLE_OFFLINE then
66:     if vm0.getRam() < vm1.getRam() then
67:         return -1
68:     end if
69:     if vm0.getRam() > vm1.getRam() then
70:         return 1
71:     end if
72:     if vm0.getMipsTotal() < vm1.getMipsTotal() then
73:         return -1
74:     end if
75:     if vm0.getMipsTotal() > vm1.getMipsTotal() then
76:         return 1
77:     end if
```

Realizadas as devidas verificações, neste ponto o algoritmo já considera que ambas as máquinas virtuais que estão sendo comparadas apresentam (i) o mesmo estado, (ii) a mesma prioridade e (iii) se estão operacionais, então os servidores que as hospedam apresentam a mesma eficiência em energia considerando tanto MIPS quanto RAM.

Na sequência, o algoritmo realiza um passo (opcional) na Linha 57. Este passo visa utilizar um estimador para calcular o tempo estimado restante para processar a máquina virtual. Apresentamos no Algoritmo 5 uma implementação de um possível estimador estatístico para esta finalidade. Neste passo uma verificação é realizada para o caso das máquinas virtuais sendo comparadas estarem operacionais, dando preferência àquela que provavelmente levará mais tempo para concluir o processamento de carga. Esta decisão é baseada em eficiência em energia, pois prioriza migrar a máquina virtual que demandará mais tempo a ser processada primeiro, podendo deixar esta alocada por mais tempo em um servidor mais eficiente em energia.

A partir da Linha 65, partimos para outros critérios de priorização de máquinas virtuais ainda não submetidas. Primeiro, tentamos dar prioridade à máquina virtual que consome menos RAM. Esta decisão é tomada com o objetivo de tentar consolidar um maior número de máquinas virtuais em servidores mais eficientes em energia. Caso ocorra empate nesta decisão, passamos a considerar o MIPS requisitado pela máquina virtual, pela mesma razão.

Prosseguindo, o algoritmo de pré-processamento verifica qual é a máquina virtual que possui o menor tamanho de arquivo de imagem. Essa decisão é tomada com base em uma heurística para minimizar o tempo médio de conclusão das cargas de trabalho,

```
78:         if vm0.getImageFilesize() < vm1.getImageFilesize() then
79:             return -1
80:         end if
81:         if vm0.getImageFilesize() > vm1.getImageFilesize() then
82:             return 1
83:         end if
84:     end if
85:     if vm0.getState() = OPERATIONAL then
86:         if vm0.getRam() < vm1.getRam() then
87:             return -1
88:         end if
89:         if vm0.getRam() > vm1.getRam() then
90:             return 1
91:         end if
92:         vm0Mips ← vm0.getMipsEffective()
93:         vm1Mips ← vm1.getMipsEffective()
94:         if vm0Mips < vm1Mips then
95:             return -1
96:         else if vm0Mips > vm1Mips then
97:             return 1
98:         end if
99:         if vm0.getWorkloadsProcessedMi()
100:                                     < vm1.getWorkloadsProcessedMi() then
101:             return -1
102:         end if
103:         if vm0.getWorkloadsProcessedMi()
104:                                      > vm1.getWorkloadsProcessedMi() then
105:             return 1
106:         end if
107:     end if
108:     return 0
109: end function
```

uma vez que máquinas com tamanhos de arquivos menores levam menos tempo para ser carregadas do local onde estão armazenadas ao servidor. Isso ajuda particularmente nos casos de imagens muito grandes em *data centers* que hospedam os arquivos de imagens em *Network Attached Storages* (NASes).

Se ocorrerem empates até este momento, consideramos ter exaurido todos os critérios que o algoritmo de pré-processamento considera relevante para o desempate para máquinas virtuais que estejam *offline*. A partir deste ponto o algoritmo faz os últimos desempates em situações nas quais as máquinas virtuais estejam *online*.

O próximo critério de desempate, visto na Linha 86, visa dar prioridade de migração às máquinas virtuais que consomem menos memória RAM. A importância deste critério é realizar migrações mais rápidas — pois teremos uma menor quantidade de dados para migrar de máquinas virtuais operacionais na migração *live* — reduzindo os impactos de desempenho para estas máquinas.

Algoritmo 10 TEA: Dead Code Estimate Time Remaining

```
1: if vm0.getState() = INOPERABLE_OFFLINE then
2:     if estimateAverageTimeRemainingToFinish(vm0)
3:                         < estimateAverageTimeRemainingToFinish(vm1) then
4:         return -1
5:     end if
6:     if estimateAverageTimeRemainingToFinish(vm0)
7:                         > estimateAverageTimeRemainingToFinish(vm1) then
8:         return 1
9:     end if
10: end if
```

A seguir, na Linha 92 comparamos o MIPS efetivo — ou seja, aquele entregue pelo servidor hospedeiro à máquina virtual. Este critério visa dar preferência à máquina virtual com menor MIPS, nessa situação, o pré-processamento objetivará a consolidação do maior número de máquinas virtuais em servidores mais eficientes em energia.

O critério final de comparação no pré-processamento é dado na Linha 100 onde verificamos a quantidade de carga que cada máquina virtual já processou. É dada prioridade à máquina virtual que tenha processado menos carga, em virtude da expectativa da máquina virtual que processou mais carga durar menos tempo, com o objetivo de reduzir o impacto no desempenho do processamento desta.

Com base nestes critérios de comparação, pode-se usar qualquer algoritmo de ordenação para definir a ordem de prioridade no conjunto das máquinas virtuais.

A versão original do algoritmo passou por diversas otimizações a fim de torná-lo mais escalável. Em princípio, pareceu ser uma boa ideia colocar no algoritmo de pré-processamento, antes da Linha 57, um passo adicional para priorizar no processo de seleção máquinas virtuais ainda não instanciadas as que potencialmente finalizarão primeiro, de modo a minimizar o tempo médio para conclusão das cargas de trabalho. No entanto, observamos que este código, com especial dependência do estimador de tempo restante para finalização das máquinas virtuais utilizado, em execução prática, para um enorme número de cenários avaliados, trouxe ganhos insignificantes, além de demandar um nível de processamento mais acentuado dos *brokers* e, por isso, foi removido do código do algoritmo de pré-processamento. Apresentamos, para apreciação, este código morto no Algoritmo 10.

6.3 `TEA` — Encontrar Servidor para uma Máquina Virtual (*FHA*)

Apresentado o pré-processamento realizado pelo *broker*, além do estimador opcional, damos início à explicação do algoritmo de escolha de servidor para hospedar uma dada máquina virtual ingressante.

O `TEA` age em pelo menos cinco critérios cruciais para determinar o melhor servidor para uma máquina virtual. São eles: (i) menor diferença de consumo de potência antes e após a alocação da máquina virtual ingressante, (ii) servidor mais eficiente em energia com

Algoritmo 11 TEA: findHostForVm(vm)

```
1:  function findHostForVm(vm)
2:      bestHost ← vm.getHostProcessing()
3:      if vm.getState() = INOPERABLE_OFFLINE then
4:          source ← getStorageForVm(vm)
5:      else
6:          source ← bestHost
7:      end if
8:      for all capable host ∈ cloud do
9:          if host = bestHost then
10:             continue
11:         end if
12:         hostEnergyEfficiencyMips ← calculateEnergyEfficiencyMips(host)
13:         if vm.getPriority() ≠ HIGH &
14:                                          hostEnergyEfficiencyMips <
    minimumAcceptableEnergyEfficiencyForGeneralUsage then
15:             continue
16:         end if
17:         hostPowerDiff ←
18:                                  estPwAfterAlloc(host, vm) - host_cur_power
19:         if hostPowerDiff < bestHost.getPowerDiff() then
20:             bestHost ← host
21:             continue
22:         else if hostPowerDiff > bestHost.getPowerDiff() then
23:             continue
24:         end if
25:         if host.energyEfficiencyMips > bestHost.energyEfficiencyMips then
26:             bestHost ← host
27:             continue
28:         else if host.energyEfficiencyMips < bestHost.energyEfficiencyMips then
29:             continue
30:         end if
```

relação ao MIPS, (iii) servidor mais eficiente em energia com relação à RAM, (iv) servidor mais eficiente em energia com relação à largura de banda e (v) maior eficiência energética com relação à largura de banda dos comutadores de pacotes da origem ao destino.

Ao receber a solicitação para determinar um servidor para uma máquina virtual vm, o *FHA* `TEA` (Algoritmo 11) inicia constatando qual servidor a está processando e, na existência deste, começa tratando-o como o melhor servidor *bestHost* encontrado até o momento para hospedá-la (Linha 2). Esta seleção visa evitar a migração de máquinas virtuais que já tenham sido alocadas no data center.

Depois, definimos a variável *source*, que representa o local onde a máquina virtual se encontra (Linha 3). Para uma máquina virtual ainda não alocada, esta variável representará o local (*storage*) onde ela está armazenada; caso ela esteja alocada em um servidor, então esta variável representará o servidor que a processa. Essa variável é usada para

```
31:         if host.energyEfficiencyRam > bestHost.energyEfficiencyRam then
32:             bestHost ← host
33:             continue
34:         else if host.energyEfficiencyRam < bestHost.energyEfficiencyRam then
35:             continue
36:         end if
37:         if host.energyEfficiencyBw > bestHost.energyEfficiencyBw then
38:             bestHost ← host
39:             continue
40:         else if host.energyEfficiencyBw < bestHost.energyEfficiencyBw then
41:             continue
42:         end if
43:         hostEnergyEfficiencyNetwork ←
44:                                 calculateEnergyEfficiencyNetwork(source, host)
45:         if hostEnergyEfficiencyNetwork
46:                                   > bestHost.getEnergyEfficiencyNetwork() then
47:             bestHost ← host
48:         end if
49:     end for
50:     return bestHost
51: end function
```

possibilitar a determinação de características da rede (e.g. rota) para transmissão — submissão ou migração — da máquina virtual.

A partir daqui, o algoritmo começa a realizar uma varredura em todos os servidores da nuvem capazes de suportar a máquina virtual com os quais o *broker* esteja programado para trabalhar (Linha 8). Entende-se um servidor como sendo capaz de suportar a máquina virtual se ele tiver condições de prover recursos suficientes à sua hospedagem — e.g. MIPS, RAM, etc. Notamos que algumas plataformas de computação em nuvem podem não dar suporte nativo à detecção de suficiência de recursos em servidores para a hospedagem de máquinas virtuais. Nestes casos, o uso de ferramentas adicionais para prover tal suporte às plataformas se faz necessário. Um exemplo deste tipo de ferramenta é a VM^2T [35].

Caso o *broker* esteja processando uma máquina virtual previamente hospedada — i.e., verificando a possibilidade de migrá-la — o servidor onde ela se encontra pode ser saltado (Linha 9) nessa verificação, pois já foi definido como sendo o *bestHost* inicialmente.

Na Linha 14 vemos em ação uma funcionalidade que objetiva prover (i) uma reserva de máquinas físicas dedicadas ao processamento de máquinas virtuais de prioridades mais elevadas e (ii) maior eficiência em energia. É permitido ao usuário do `TEA` definir um parâmetro *minimumAcceptableEnergyEfficiencyForGeneralUsage*, que representa a eficiência em energia mínima aceitável que um servidor deve ter para hospedar máquinas virtuais de prioridades que não sejam altas. Em outras palavras, podemos deixar o escalonamento mais eficiente em energia ou com melhor desempenho facilmente modificando este parâmetro. Podemos, por exemplo, fazer com que o `TEA` seja executado em modo econômico (`TEAe`) especificando esta variável para que o `TEA` utilize no máximo 50%

dos servidores para hospedarem máquinas virtuais que não tenham prioridades elevadas; podemos fazer com que o `TEA` seja executado voltado para máximo desempenho (`TEAp`) configurando-o para utilizar 100% dos servidores no processo de escalonamento; ou podemos fazer uma operação balanceada configurando o `TEA` para utilizar no máximo 75% dos servidores para essa finalidade (`TEAb`). Vale ressaltar que caso o `TEA` esteja programado para operar utilizando um percentual muito elevado (e.g. 100%) dos servidores, e a nuvem esteja saturada, e ingressar máquinas virtuais de alta prioridade, então estas poderão ter sua inicialização postergadas, quando comparadas aos casos mais econômicos.

Seja lee_{dec} uma lista ordenada de modo decrescente por eficiência em energia dos servidores que compõem a nuvem, lee_{size} o tamanho desta lista, e $general\_hosts_{\%}$ o valor percentual desejado. Com estes dados, podemos usar a Função (6.3), que recebe como entrada o percentual de servidores para uso geral, para calcular facilmente o valor de $minimumAcceptableEnergyEfficiencyForGeneralUsage$. Observe que, por essa fórmula, quanto maior for o número de servidores para uso geral, menor será a eficiência em energia exigida.

$$\texttt{calculateMinimumAcceptableEnergyEfficiencyForGeneralUsage}(general\_hosts_{\%}) = lee_{dec}\texttt{.get(round((}lee_{size} - 1\texttt{)} \times general\_hosts_{\%}\texttt{))} \quad (6.3)$$

A seguir, o algoritmo usa como critério de comparação a diferença de potência consumida estimada antes e após a alocação da máquina virtual (Linha 18). Neste critério compara-se a diferença do incremento do consumo de potência acarretado pela alocação de *vm* em *host* com o de *vm* alocada em *bestHost*. Se *vm* alocada em *host* tiver uma menor variação do consumo de potência do que *vm* alocada em *bestHost*, então *bestHost* é atualizado. Caso *bestHost* não esteja definido, então a função `estPwAfterAlloc()` (Algoritmo 7) retornará o valor ∞. Caso a máquina esteja previamente hospedada na máquina a qual deseja se calcular a diferença de energia do servidor com e sem a máquina virtual, a função `estimatePowerAfterDeallocation()` (8) poderá ser utilizada. O método `getPowerDiff()` calcula a diferença de potência consumida por *bestHost* de acordo com o caso aplicável.

A seguir (Linha 25), o `TEA` verifica se a eficiência em energia do servidor *host* é melhor do que o melhor servidor encontrado *bestHost*. Caso afirmativo, ele atualiza o *bestHost* como sendo *host* e ignora as demais comparações e continua a iteração. Caso a eficiência em energia de *host* seja pior do que *bestHost*, então não compensa atualizar e nem verificar os critérios adiante, portanto nesta situação ele também ignora as demais comparações e continua a iteração.

Todas as comparações a seguir também ocorrem de modo análogo à eficiência em energia comparando o MIPS: (i) se *host* for melhor do que *bestHost* no critério valiado então *bestHost* é substituído por *host*, não são feitas mais comparações e a iteração continua; (ii) se *host* for pior do que *bestHost* no critério avaliado então não são feitas mais comparações e a iteração continua; (iii) se empatarem no quesito corrente então o próximo quesito passa a ser avaliado.

A próxima avaliação é feita comparando a eficiência em energia com base na RAM (Linha 31), e na sequência compara-se a eficiência em energia com base na largura de banda (Linha 37). A eficiência em energia de todos estes recursos é feita com base na Equação (6.1).

A última avaliação realizada (Linha 46) é um pouco mais complexa. Nela, objetivamos calcular a eficiência em energia da rede de um sistema final de origem *source*, onde a máquina virtual está hospedada, até um sistema final de destino *host*, para onde ela deverá ser submetida. Para calcular a eficiência em energia da rede — rota — entre *source* e *host*, empregamos, para cada comutador de pacotes intermediário, pertencente à nuvem, a Equação (6.1), definindo como recurso a largura de banda máxima entre origem e destino. A fórmula para calcular a eficiência em energia da rede $network\_ee$ segue a apresentada no Algoritmo 6, mas podemos também escrever de forma simplificada a eficiência em energia entre *source* e *host* como na Equação (6.4).

$$network\_ee(source, destination) = \frac{max\_bw(source, host)}{\sum_{\forall ports \in route} port.getOwner()_{max\_power}} \tag{6.4}$$

Nessa equação, a largura de banda máxima entre um sistema final de origem *source* e um sistema final de destino *destination* como apresentado na Equação (6.5). Nesta equação, $max\_bw(source, destination)$ representa a largura de banda máximo entre um sistema final de origem *source* e um sistema final de destino *destination*; *port* representa cada porta da rota *route* pela qual os pacotes deverão trafegar entre *source* e *destination*; *bw* representa a largura de banda do trajeto do tráfego na porta em questão, sendo referente ao *upload* se o sentido do tráfego for de envio ou *download* se o sentido do tráfego na porta for de download.

$$max\_bw(source, destination) = \min\{\forall ports \in route\ bw\} \tag{6.5}$$

A ordem das comparações/critérios usados no *FHA* foi definida após uma extensa experimentação, com o objetivo de maximizar a eficiência em energia.

Nós tentamos diversos outros recursos e critérios na implementação do *FHA* `TEA`, com o objetivo de torná-lo mais eficiente em energia. No entanto, tais critérios foram removidos do algoritmo. Tratam-se de recursos que em princípio pareciam fazer sentido serem utilizados, mas que quando colocados em teste — na avaliação prática por uma extensa simulação para um grande número de cenários — constatamos que eles não compensavam — seja por prejudicar a escalabilidade a ponto de se tornarem implausíveis e/ou não trazerem benefícios consideráveis. A seguir, elencamos alguns destes recursos.

Um recurso que tentamos utilizar foi evitar migrar máquinas virtuais caso o tempo estimado para a migração das máquinas virtuais fosse maior do que o tempo restante estimado para a finalização destas. Considerando o contexto de migrações pré-cópia, apresentamos na Equação (6.6) uma fórmula simplificada para estimar o tempo de migração de uma máquina virtual de um servidor de origem *source* para um servidor de destino *destination*. Nesta fórmula, $estimated\_migration\_time$ representa o tempo estimado para a migração, vm_{ram} representa a quantidade total de RAM que a máquina

virtual vm ocupa, e $estimated\_bw_{source\_destination}$ representa a largura de banda disponível estimada entre *source* e *destination* para realização da migração. Este critério se mostrou inadequado por vários motivos, em especial no contexto da grande escala: (i) o custo computacional para analisar a largura de banda estimada entre dois sistemas finais em uma rede complexa é grande, (ii) precisa-se ter um mecanismo de alta complexidade para que, uma vez a migração iniciada, não se modifique a largura de banda em nenhum trajeto da topologia entre os sistemas que participam da migração; (iii) o item (ii) tem como consequência travar a rede para possíveis novas submissões de máquinas virtuais e cargas de trabalho elevadas; (iv) o estimador da largura de banda disponível entre *source* e *destination* pode ter uma complexidade muito alta e falhar; (v) esta é uma estimativa simplificada e pode falhar consideravelmente em ambientes reais, em especial no contexto de migrações pós-cópia em cenários com grandes alterações no conteúdo da RAM da máquina virtual em questão.

$$estimated\_migration\_time = \frac{vm_{ram}}{estimated\_bw_{source\_destination}} \tag{6.6}$$

Outro critério que foi removido no processo de otimização do `TEA`, foi o de tentar dar preferência explícita para rotas que apresentem o maior número de comutadores de pacotes ligados entre o sistema final de origem e o sistema final de destino da máquina virtual. Este critério foi removido, pois (i) a análise destes percursos consome um processamento considerável em contextos de escalas maiores e (ii) na prática isso já tende a ocorrer em virtude do critério de desempate por maximização da eficiência em energia da rota, presente na Linha 46 do Algoritmo 11.

Mais um critério que não entrou no `TEA` foi a minimização do número de saltos entre o sistema final de origem e o sistema final de destino da máquina virtual. Este critério não foi incluído pois (i) a análise destes percursos consome um processamento considerável em contextos de escalas maiores e (ii) o cálculo da eficiência em energia da rota, presente na Linha 46 do Algoritmo 11, contempla, implicitamente, a maior parte dos benefícios que poderiam ser providos por utilizar este critério.

Foi descartado, também, no `TEA` a utilização de rotas dedicadas para as conexões (de submissão/migração de máquinas virtuais). Embora este critério, em princípio, tenha parecido bom no sentido de ter potencial para minimizar o tempo médio para conclusão de máquinas virtuais, na prática, em grande escala, (i) apresenta uma complexidade elevada, (ii) trava partes da topologia, (iii) em consequência a (ii) inviabiliza uma quantidade de servidores eficientes em energia, acessíveis através de partes da rede pelas quais passam tráfego, de receberem máquinas virtuais. Para ilustrar este problema, damos um exemplo: considere, por exemplo, duas rotas, (a) e (b), distintas, com partes coincidentes. Considere que uma imagem de máquina virtual de 8 GB começa a ser transmitida pela rota (a) e, posteriormente, seria desejável para o algoritmo a submissão de uma outra máquina virtual, de 500 MB, pela rota (b). No entanto, esta segunda máquina virtual vai precisar ser postergada — ou pior, precisar usar outra rota para outro sistema final de destino, potencialmente menos eficiente em energia — em virtude da ocupação da rota, ainda que tenha um tamanho muito menor do que a primeira. O mesmo ocorre para migrações de máquinas virtuais com diferenças de tamanhos de RAMs significativas. Além disso, uma

solução de complexidade baixa e satisfatória, indicada para o uso do `TEA` — e também de outros algoritmos de escalonamento — é a limitação do número de conexões simultâneas para submissão ou migração de máquinas virtuais.

Outro critério que desconsideramos para o `TEA` foi o de dar preferência aos servidores que já estão hospedando maior número de máquinas virtuais. A ideia inicial deste critério era que, um servidor hospedando um grande número de máquinas virtuais tenderá a permanecer ligado em virtude da consolidação, evitando migrações. No entanto, avaliando os diversos cenários estudados percebemos que o número de migrações que este recurso evita é desprezível, pois via de regra compensa realizar migrações quando existem outros servidores mais eficientes em energia.

No mesmo sentido da ideia e da dispensa do critério anterior, dispensamos também dar preferência aos servidores que possuem maior quantidade de RAM. Mais do que isso, usualmente o critério presente na Linha 31 do Algoritmo 11 engloba, indiretamente, na maioria dos casos, este desempate.

Um outro fator que foi descartado também foi o de menor latência estimada entre o sistema final de origem e o sistema final de destino da máquina virtual. Este critério foi descartado, pois (i) minimizar latência gera impacto pontual com arquivos grandes como são os casos das imagens de máquinas virtuais ou com a grande quantidade de dados que é necessário transmitir durante a migração de uma máquina virtual, além (ii) da complexidade envolvida para estimar essa latência. Em outras palavras, a estimativa da soma das estimativas de atraso nodal, atraso de enfileiramento, atraso de propagação e atraso de transmissão por toda a topologia, além de consumir um processamento considerável em um processo complexo, trás um benefício muito pequeno para grandes quantidades de dados — o *overhead* por atraso é muito pequeno nesta situação.

6.4 Simulação

Adotamos a simulação como método para validação e possibilitar a análise de resultados. Por razões históricas e de aproveitamento de códigos preexistentes, inicialmente cogitamos utilizar o *CloudSim*. No entanto, observamos severas limitações nesse simulador, tais como: inexistência de suporte oficial ao consumo de energia no núcleo da rede; questões técnicas relatadas em fóruns mostrando a dificuldade elevada de prover tal suporte nativamente; questões de escalabilidade — abrangência muito larga para um contexto IaaS que torna as simulações pesadas — para um grande número de simulações necessárias para a avaliação desejável do `TEA`; funcionamento padrão de não usar *storages* para armazenar as imagens de máquinas virtuais; modelos de potência severamente limitados — ex: não consideram tempos de ligamento/desligamento e nem o processamento durante este tempo de servidores e elementos de rede; incapacidade de avaliar nativamente tanto o tempo para conclusão de processamento de cargas quanto o tempo de processamento de cargas — mensurado a partir do recebimento da carga pela máquina virtual; entre diversas outras questões.

Por este motivo, consideramos uma substituição de simulador e, dentre uma gama de outros simuladores, consideramos o mais adequado para essa simulação o *Sinergy-Cloud* [25], descrito com mais detalhes no Capítulo A.

6.4.1 Parâmetros das Simulações

Nas topologias geradas com limitação do número de portas dos *switches*, cada *switch* possui um limite superior de 24 portas.

Consideramos as seguintes possibilidades para compôr os cenários de simulação: *data centers* geodistribuídos e não geodistribuídos; com topologias *Single-Hop Star*, *Flat-Tree*, *Binary Tree* e *K-Ary Fat Tree*; com migrações ativadas e desativadas; homogêneos e heterogêneos; com nuvens constituídas por 10, 100 e 1.000 servidores; com SLA de garantia de 100% de processamento ativado e desativado; comportamento de processamento de usuários seguindo um padrão histórico ou não; máquinas virtuais com somente uma ou com múltiplas prioridades e as migrações podem ser executadas quando ocorrem atualizações do *broker*, estas definidas para ocorrer nos seguintes eventos: inicialização do *broker*, inicialização de máquina virtual, finalização de máquina virtual, inicialização de carga de trabalho, finalização de carga de trabalho, finalização de migração de máquina virtual e abortamento de migração de máquina virtual.

Os algoritmos de escalonamento de máquinas virtuais selecionados para as comparações foram o *Round-Robin* (`RR`), *Random* (`RND`), *First Available* (`FA`), *Best Resource Selection — Highest Utilization* (`HU`), *Best Resource Selection — Highest Requisition* (`HR`), *Minimum Power Diff* (`MPD`), *Lago Allocator* (`LA`), *Bandwidth-Aware Lago Allocator* (`BALA`) e seis sabores do `TEA`, a seguir elencadas:

- `TEAp` → `TEA` **P**erfomance, `TEA` com utilização de 100% dos servidores para uso geral e sem estimador de tempo de finalização de máquinas virtuais;
- `TEApf` → `TEA` **P**erfomance com estimador de tempo de **F**inalização de máquinas virtuais;
- `TEAb` → `TEA` **B**alanced, `TEA` com utilização de 75% dos servidores para uso geral e sem estimador de tempo de finalização de máquinas virtuais;
- `TEAbf` → `TEA` **B**alanced com estimador de tempo de **F**inalização de máquinas virtuais;
- `TEAe` → `TEA` **E**co, `TEA` com utilização de 50% dos servidores para uso geral e sem estimador de tempo de finalização de máquinas virtuais;
- `TEAef` → `TEA` **E**co com estimador de tempo de **F**inalização de máquinas virtuais.

Observamos que os algoritmos de escalonamento de máquinas virtuais utilizados para comparação usualmente são empregados em um *data center*, por um *broker*, para determinar um servidor físico, dentro deste *data center*, para hospedar a máquina virtual ingressante. Portanto, este cenário clássico não lida com a geodistribuição de *data centers*.

Para lidar com esta limitação em cenários geodistribuídos, implementamos estes algoritmos clássicos para selecionar servidores não somente dentro de um *data center*, mas da nuvem como um todo.

Contemplamos todos os conjuntos de possibilidades apresentadas em cenários. Cada cenário foi executado 30 vezes para cada algoritmo de escalonamento de máquinas virtuais. Ou seja, todas as possibilidades de simulações para quatro modelos de topologias, duas possibilidades quanto o suporte à migração de máquinas virtuais, duas possibilidades quanto ao uso de SLA, duas possibilidades de uso quanto à definição do comportamento de usuários, duas possibilidades quanto ao uso de prioridades, duas possibilidades quanto ao uso de geodistribuição, quatorze algoritmos de escalonamento, dois tipos de *data centers* e três tamanhos de *data centers*, resultando um total de $4 \times 2 \times 2 \times 2 \times 2 \times 2 \times 14 \times 2 \times 3 = 10.752$ cenários simulados, portanto, um total de 322.560 simulações realizadas foram consideradas.

Consideramos para cada cenário uma relação 1:4 entre número de servidores e número de máquinas virtuais; bem como 1:4 entre número de usuários e número de máquinas virtuais — e, por consequência, o número de usuários é idêntico ao número de servidores que compõem a nuvem — cada máquina virtual recebe uma carga de trabalho; e a imagem de cada máquina virtual é armazenada em um *storage* na proporção 35:1, ou seja, cada *storage* fica responsável pelo armazenamento de 35 imagens de máquinas virtuais. Por questões de maximização de desempenho e para uma comparação justa, o número de submissões e o número de migrações de máquinas virtuais é igual ao número de *storages* — isso evita, por exemplo, sobrecarga da rede por excesso de migrações ou um número muito grande de conexões simultâneas de envio de máquinas virtuais a partir de um mesmo *storage*, acarretando em um aumento de tempo considerável para transmissão das imagens das máquinas virtuais.

Consideramos os tempos de ligamento de servidores, desligamento de servidores, ligamentos de *switches* e desligamento de *switches* para serem, respectivamente, 20, 10, 10 e 5 segundos. Configuramos cada *switch* para ser desligado automaticamente após 30 minutos sem atividade de conexão, e cada servidor para ser desligado após 30 minutos de ociosidade. Se este valor for muito baixo, então teríamos sucessivos desligamentos e religamentos, o que acarretaria em um *overhead* muito grande; mas se for muito alto então não conseguiríamos uma boa eficiência em energia. Consideramos que servidores e *switches* podem ser desligados, enquanto roteadores de borda, *brokers* e *storages* não — ou seja, estes estão permanentemente ligados.

Cada *storage* foi definido para ter 10 Gbps de largura de banda e 4 kTB de espaço de armazenamento. O modelo de potência usado nos *storages* é o DVFS linear, com consumo mínimo de 175 W e máximo de 250 W, e desligamento desativado.

As nuvens não geodistribuídas são constituídas por um *data center* onde ficam todos os sistemas finais, as nuvens geodistribuídas são constituídas por dois *data centers* com distribuição igualitária dos sistemas finais entre os *data centers*. As taxas de transmissão entre os *data centers* geodistribuídos é de 100 Mbps.

Nos *data centers* homogêneos, consideramos todos os servidores possuindo a mesma configuração: 1 Gbps de largura de banda; 16 GiB de RAM; processador i7 6700K ($4 \times$

4.000 MHz); e um modelo de potência DVFS linear com consumo máximo de pico de 400 W.

Nos *data centers* heterogêneos, adotamos as seguintes configurações para os servidores:

- Largura de banda de 100 Mbps, 1 Gbps e 10 Gbps, em distribuição *round-robin*;
- 16 GiB de RAM por servidor;
- Processadores baseados nos modelos Xeon E3-1505M v6 (4×3.000 MHz), Xeon E7-8893 v4 (4×3.500 MHz) e i7 6700K (4×4.000 MHz), em distribuição *round-robin*;
- Modelos de potência DVFS linear, baseado nas máquinas HP DL380, IBM BladeCenter H Blade Server, Dell PowerEdge R710, IBM i5-520 e Sun V490, com consumos máximos de, respectivamente, 262, 320, 570, 582 e 651 W, e mínimos de 70% deste valor; com distribuição *round-robin*.

Para poder avaliar de forma mais dedicada os efeitos da ciência da topologia no processo do escalonamento — para instanciação e migração de máquinas virtuais — consideramos pontual o consumo de taxa de transmissão utilizada pelos sistemas em operação das máquinas virtuais e que os servidores não rodam nenhum tipo de *EBPA*, ou seja, inexiste a reserva de largura de banda dedicada ou extra para máquinas virtuais.

A distribuição do armazenamento das imagens das máquinas virtuais pelos *storages* é feita em *round-robin*.

A colocação dos *switches* obedece distribuição *round-robin*, sendo que cada *switch* pode possuir todas as portas com largura de banda de 1 Gbps ou todas as portas com largura de banda de 10 Gbps. Estes valores foram escolhidos intencionalmente para tentar mostrar a importância em conhecer as larguras de banda dos elementos que compõem o núcleo da topologia — e não somente os elementos de borda. Por consequência, esta decisão de implementação afeta de forma negativa algoritmos que consideram somente a largura de banda dos elementos de borda, como o `BALA`.

O modelo de potência dos *switches* foi o DVFS linear e baseado nos modelos Cisco SPS2024, Cisco SGE2000P e NETGEAR ProSAFE M7100-24X, com consumo máximo respectivamente de 33, 102 e 195,2 W, e mínimos a 70% deste valor; com distribuição *round-robin*.

Os roteadores de borda apresentam interface interna de 10 Gbps e modelo de potência DVFS linear de 120 W, baseado no modelo Cisco 2801.

A pilha de protocolos utilizada foi constituída de *Ethernet*, IPv6 e TCP.

As máquinas virtuais foram inspiradas nos modelos do Amazon AWS m3.medium, m4.large e t2.medium, possuindo, processadores com núcleos $\times$ *clock*, respectivamente, 1×2.500, 2×2.400 e 2×3.300 em distribuição *round-robin*. Seguindo esta inspiração, a RAM obedece a distribuição *round-robin* com tamanhos de 4 GiB e 8 GiB.

Nos cenários com somente uma prioridade, todas as máquinas virtuais apresentam prioridade normal. Nos cenários com múltiplas prioridades, é feita a seguinte distribuição *round-robin* entre as máquinas virtuais: alta, alta, normal, normal, normal, baixa, baixa.

As imagens das máquinas virtuais foram baseadas nas atuais distribuições Linux Debian, Ubuntu Server CLI, OpenSUSE, Ubuntu Server e Fedora, possuindo, respectivamente, 500 MB, 1,5 GB, 3 GB, 5 GB e 10 GB.

As cargas de trabalho têm tamanho médio de 12 milhões de MI, sendo aleatório no intervalo de 2 milhões – 72 milhões de MI. Nos cenários sem comportamento de usuário definido esta distribuição é totalmente aleatória, e nos cenários com comportamento de usuário definido essa distribuição é feita de modo que cada as cargas de trabalho tenham uma discrepância máxima de ± 50% em relação à média das cargas por usuário.

Por fim, o *broker* apresenta largura de banda de 10 Gbps, modelo de potência DVFS linear com consumo mínimo de 175 W e máximo de 250 W.

A seguir, iniciamos a análise dos resultados obtidos pela simulação. Em prol da síntese e considerando a enorme gama de possibilidades abordadas para tornar este trabalho completo, optamos por iniciar apresentando uma característica representativa do *data center* por vez ao invés de abordar todos os cenários simulados. Nos histogramas referentes ao consumo de energia, não contemplamos os algoritmos `RR` e `RND`, pois como se podia esperar estes algoritmos — não cientes de energia — apresentaram valores de consumo muito alto, prejudicando a boa visualização dos resultados. Cada entrada na tabela, portanto, se refere ao número total de cenários (no caso, 10.752) dividido pelo número de valores possíveis para a característica avaliada: exemplo (i), geodistribuição de *data centers* pode assumir dois valores: geodistribuídos ou não geodistribuídos, logo cada linha para este critério abrange 10.752/2 = 5.376 cenários; exemplo (ii) a nuvem pode apresentar três tamanhos, com 10, 100 e 1.000 servidores, logo cada linha para este critério abrange 10.752/3 = 3.584 cenários. Resultados consolidados serão apresentados mais à frente.

6.4.2 Análise de Resultados: Energia Total

A Figura 6.1 traz informações sobre a energia total consumida pela nuvem. Neste cenário, avaliamos uma nuvem possuindo 10 servidores distribuídos em dois *data centers* geodistribuídos e heterogêneos, empregando topologia *Flat-Tree*. O recurso de migração de máquinas virtuais está ativado, com SLA de garantia de 100% de processamento para as máquinas virtuais, com comportamento de usuários indefinido e cargas de trabalho com somente uma prioridade. O algoritmo de escalonamento que apresentou melhores resultados (18,49 ± 0,22 kWh) para este quesito foi o `TEA` (`TEAe`), apresentando, considerando o intervalo de confiança, isoladamente os melhores valores do parâmetro comparado dentre todos os algoritmos para este cenário, com uma melhora variando de 15,9% a 21,8% comparado ao segundo melhor (`BALA`).

Apresentamos na Figura 6.2 uma comparação da energia total consumida pela nuvem. Neste cenário, avaliamos uma nuvem possuindo 1.000 servidores distribuídos em dois *data centers* geodistribuídos e heterogêneos, empregando topologia *K-Ary Fat Tree*. O recurso de migração de máquinas virtuais está desativado, com SLA de garantia de 100% de processamento para as máquinas virtuais, com comportamento de usuários definido e cargas de trabalho com somente uma prioridade. O algoritmo de escalonamento que apresentou melhores resultados (2.753,38 ± 30,65 kWh) para este quesito foi o `TEA`

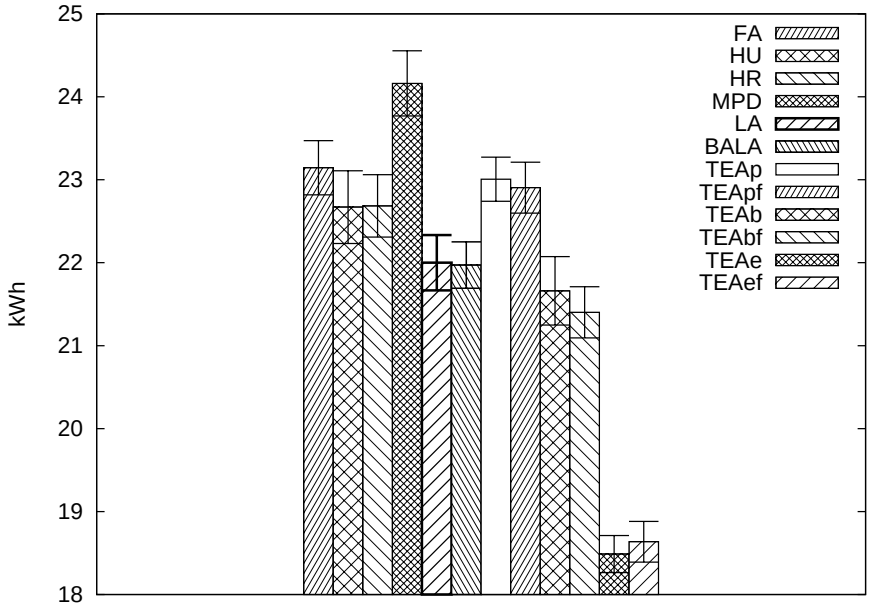

Figura 6.1: Energia Total: DC Geo. Het., 10 srvs, Flat-Tree, Mig, SLA, Comp Usr Ind, S/Mult Pr

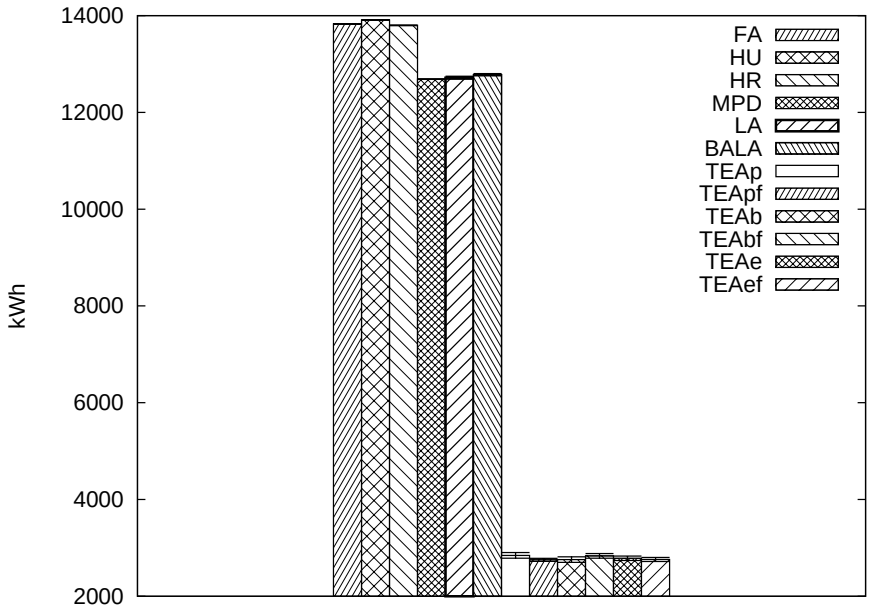

Figura 6.2: Energia Total: DC Geo. Het., 1.000 srvs, K-Ary Fat Tree, S/Mig, SLA, Comp Usr Def, S/Mult Pr

(`TEApf`), apresentando, considerando o intervalo de confiança, isoladamente os melhores valores do parâmetro comparado dentre todos os algoritmos para este cenário, com uma melhora variando de 355,5% a 366,1% comparado ao segundo melhor (`MPD`).

Apresentamos na Figura 6.3 uma comparação da energia total consumida pela nuvem. Neste cenário, avaliamos uma nuvem possuindo 100 servidores distribuídos em dois *data centers* geodistribuídos e homogêneos, empregando topologia *K-Ary Fat Tree*. O recurso de migração de máquinas virtuais está desativado, sem SLA de garantia de 100% de pro-

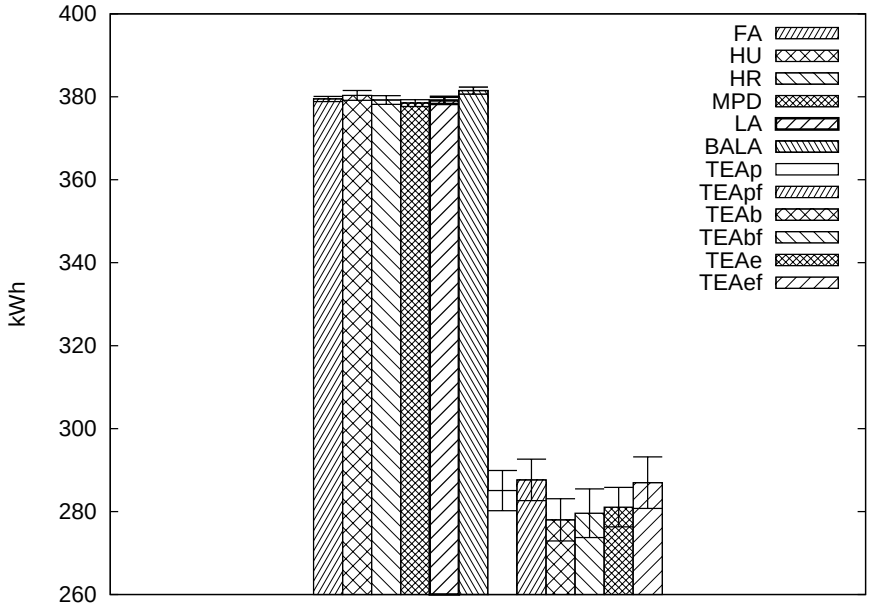

Figura 6.3: Energia Total: DC Geo. Homo., 100 srvs, K-Ary Fat Tree, S/Mig, S/SLA, Comp Usr Def, Mult Pr

cessamento para as máquinas virtuais, com comportamento de usuários definido e cargas de trabalho com múltiplas prioridades. O algoritmo de escalonamento que apresentou melhores resultados (278,00 ± 5,08 kWh) para este quesito foi o `TEA` (`TEAb`), apresentando, considerando o intervalo de confiança, isoladamente os melhores valores do parâmetro comparado dentre todos os algoritmos para este cenário, com uma melhora variando de 33,4% a 39,0% comparado ao segundo melhor (`MPD`).

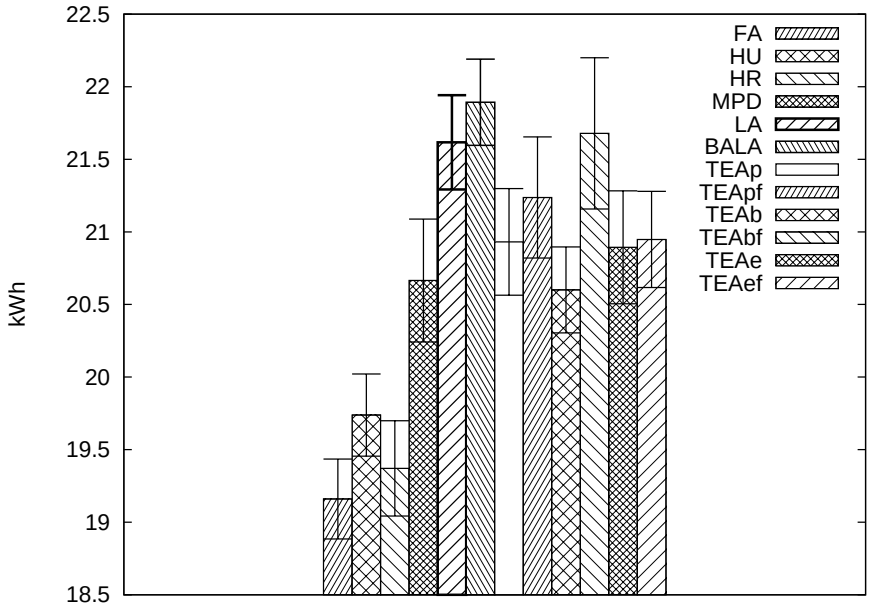

Figura 6.4: Energia Total: DC Geo. Homo., 10 srvs, Single-Hop Star, Mig, SLA, Comp Usr Def, Mult Pr

Apresentamos na Figura 6.4 uma comparação da energia total consumida pela nuvem. Neste cenário, avaliamos uma nuvem possuindo 10 servidores distribuídos em dois *data centers* geodistribuídos e homogêneos, empregando topologia *Single-Hop Star*. O recurso de migração de máquinas virtuais está ativado, com SLA de garantia de 100% de processamento para as máquinas virtuais, com comportamento de usuários definido e cargas de trabalho com múltiplas prioridades. O algoritmo de escalonamento que apresentou melhores resultados (19,16 ± 0,28 kWh) para este quesito foi o `FA`, empatado tecnicamente pelo menos com o segundo melhor, `HR` (19,37 ± 0,33 kWh).

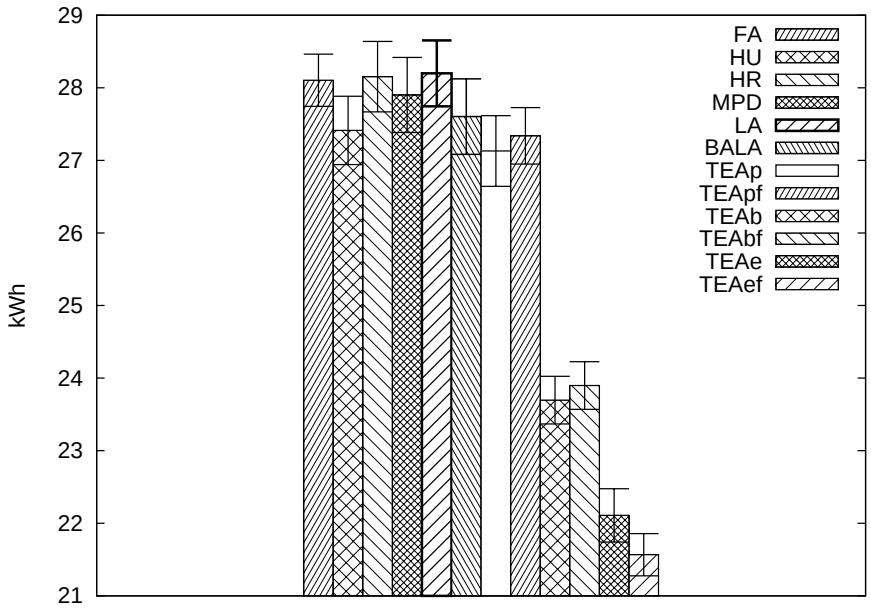

Figura 6.5: Energia Total: DC N.Geo. Het., 10 srvs, Binary Tree, S/Mig, SLA, Comp Usr Def, Mult Pr

Mostramos na Figura 6.5 um gráfico da energia total consumida pela nuvem. Neste cenário, avaliamos uma nuvem possuindo 10 servidores distribuídos em um *data center* não geodistribuído e heterogêneo, empregando topologia *Binary Tree*. O recurso de migração de máquinas virtuais está desativado, com SLA de garantia de 100% de processamento para as máquinas virtuais, com comportamento de usuários definido e cargas de trabalho com múltiplas prioridades. O algoritmo de escalonamento que apresentou melhores resultados (21,57 ± 0,29 kWh) para este quesito foi o `TEA` (`TEAef`), apresentando, considerando o intervalo de confiança, isoladamente os melhores valores do parâmetro comparado dentre todos os algoritmos para este cenário, com uma melhora variando de 23,3% a 31,1% comparado ao segundo melhor (`HU`).

A Figura 6.6 traz informações sobre a energia total consumida pela nuvem. Neste cenário, avaliamos uma nuvem possuindo 10 servidores distribuídos em um *data center* não geodistribuído e heterogêneo, empregando topologia *Flat-Tree*. O recurso de migração de máquinas virtuais está ativado, sem SLA de garantia de 100% de processamento para as máquinas virtuais, com comportamento de usuários indefinido e cargas de trabalho com somente uma prioridade. O algoritmo de escalonamento que apresentou melhores resultados (14,42 ± 0,13 kWh) para este quesito foi o `TEA` (`TEAe`), apresentando, considerando

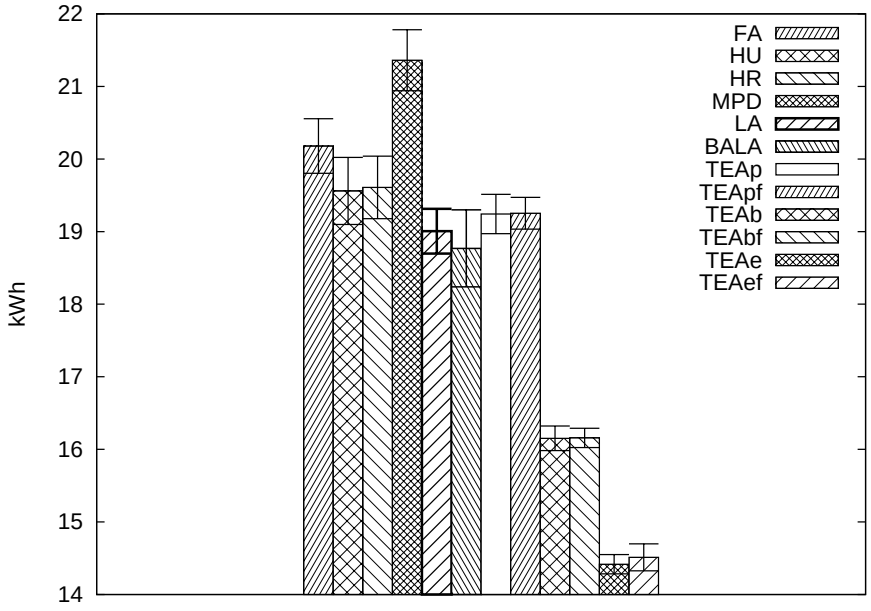

Figura 6.6: Energia Total: DC N.Geo. Het., 10 srvs, Flat-Tree, Mig, S/SLA, Comp Usr Ind, S/Mult Pr

o intervalo de confiança, isoladamente os melhores valores do parâmetro comparado dentre todos os algoritmos para este cenário, com uma melhora variando de 25,4% a 35,1% comparado ao segundo melhor (`BALA`).

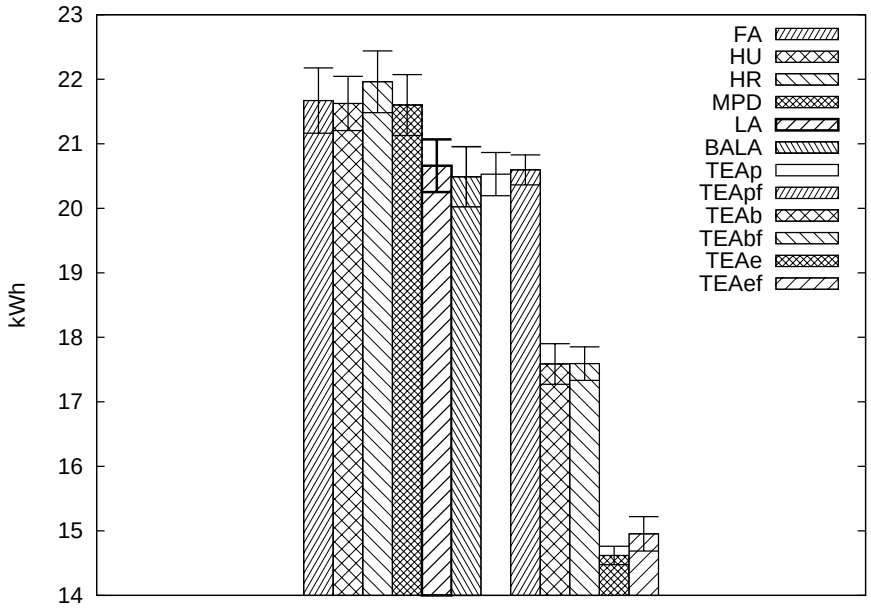

Figura 6.7: Energia Total: DC N.Geo. Het., 10 srvs, Flat-Tree, S/Mig, SLA, Comp Usr Def, Mult Pr

A Figura 6.7 traz informações sobre a energia total consumida pela nuvem. Neste cenário, avaliamos uma nuvem possuindo 10 servidores distribuídos em um *data center* não geodistribuído e heterogêneo, empregando topologia *Flat-Tree*. O recurso de migração de

máquinas virtuais está desativado, com SLA de garantia de 100% de processamento para as máquinas virtuais, com comportamento de usuários definido e cargas de trabalho com múltiplas prioridades. O algoritmo de escalonamento que apresentou melhores resultados (14,62 ± 0,14 kWh) para este quesito foi o `TEA` (`TEAe`), apresentando, considerando o intervalo de confiança, isoladamente os melhores valores do parâmetro comparado dentre todos os algoritmos para este cenário, com uma melhora variando de 35,6% a 44,7% comparado ao segundo melhor (`BALA`).

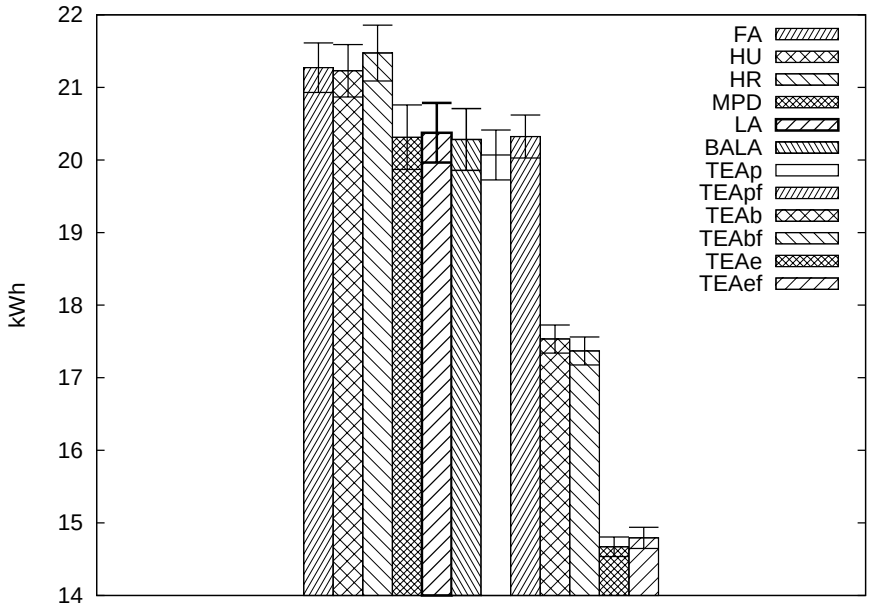

Figura 6.8: Energia Total: DC N.Geo. Het., 10 srvs, Flat-Tree, S/Mig, SLA, Comp Usr Ind, Mult Pr

Temos na Figura 6.8 um histograma sobre a energia total consumida pela nuvem. Neste cenário, avaliamos uma nuvem possuindo 10 servidores distribuídos em um *data center* não geodistribuído e heterogêneo, empregando topologia *Flat-Tree*. O recurso de migração de máquinas virtuais está desativado, com SLA de garantia de 100% de processamento para as máquinas virtuais, com comportamento de usuários indefinido e cargas de trabalho com múltiplas prioridades. O algoritmo de escalonamento que apresentou melhores resultados (14,67 ± 0,13 kWh) para este quesito foi o `TEA` (`TEAe`), apresentando, considerando o intervalo de confiança, isoladamente os melhores valores do parâmetro comparado dentre todos os algoritmos para este cenário, com uma melhora variando de 34,1% a 42,5% comparado ao segundo melhor (`BALA`).

Apresentamos na Figura 6.9 uma comparação da energia total consumida pela nuvem. Neste cenário, avaliamos uma nuvem possuindo 10 servidores distribuídos em um *data center* não geodistribuído e heterogêneo, empregando topologia *K-Ary Fat Tree*. O recurso de migração de máquinas virtuais está desativado, com SLA de garantia de 100% de processamento para as máquinas virtuais, com comportamento de usuários definido e cargas de trabalho com múltiplas prioridades. O algoritmo de escalonamento que apresentou melhores resultados (23,54 ± 0,39 kWh) para este quesito foi o `TEA` (`TEAe`), apresentando, considerando o intervalo de confiança, isoladamente os melhores valores do parâmetro

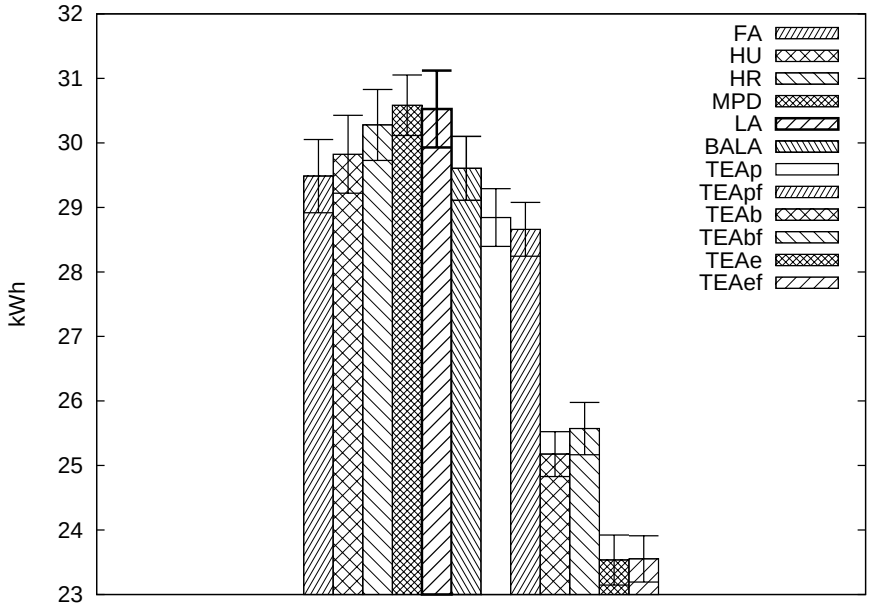

Figura 6.9: Energia Total: DC N.Geo. Het., 10 srvs, K-Ary Fat Tree, S/Mig, SLA, Comp Usr Def, Mult Pr

comparado dentre todos os algoritmos para este cenário, com uma melhora variando de 20,9% a 29,8% comparado ao segundo melhor (`FA`).

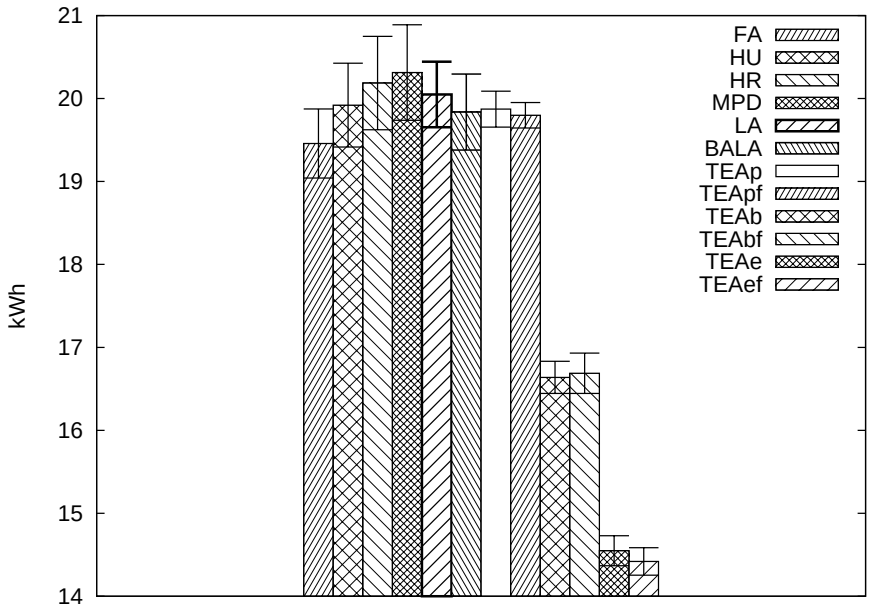

Figura 6.10: Energia Total: DC N.Geo. Het., 10 srvs, Single-Hop Star, S/Mig, S/SLA, Comp Usr Ind, S/Mult Pr

Mostramos na Figura 6.10 um gráfico da energia total consumida pela nuvem. Neste cenário, avaliamos uma nuvem possuindo 10 servidores distribuídos em um *data center* não geodistribuído e heterogêneo, empregando topologia *Single-Hop Star*. O recurso de migração de máquinas virtuais está desativado, sem SLA de garantia de 100% de processa-

mento para as máquinas virtuais, com comportamento de usuários indefinido e cargas de trabalho com somente uma prioridade. O algoritmo de escalonamento que apresentou melhores resultados (14,42 ± 0,17 kWh) para este quesito foi o `TEA` (`TEAef`), apresentando, considerando o intervalo de confiança, isoladamente os melhores valores do parâmetro comparado dentre todos os algoritmos para este cenário, com uma melhora variando de 30,6% a 39,4% comparado ao segundo melhor (`FA`).

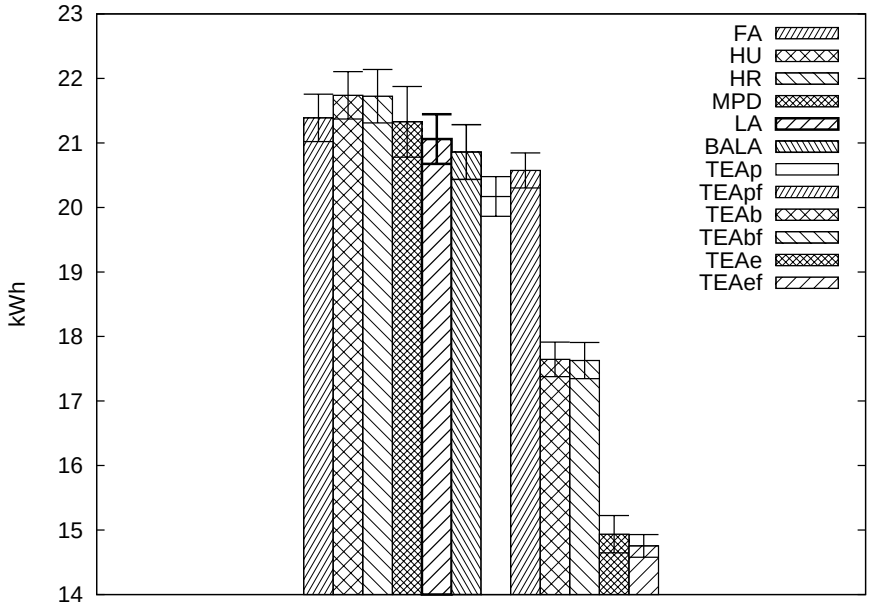

Figura 6.11: Energia Total: DC N.Geo. Het., 10 srvs, Single-Hop Star, S/Mig, SLA, Comp Usr Def, Mult Pr

Temos na Figura 6.11 um histograma sobre a energia total consumida pela nuvem. Neste cenário, avaliamos uma nuvem possuindo 10 servidores distribuídos em um *data center* não geodistribuído e heterogêneo, empregando topologia *Single-Hop Star*. O recurso de migração de máquinas virtuais está desativado, com SLA de garantia de 100% de processamento para as máquinas virtuais, com comportamento de usuários definido e cargas de trabalho com múltiplas prioridades. O algoritmo de escalonamento que apresentou melhores resultados (14,75 ± 0,18 kWh) para este quesito foi o `TEA` (`TEAef`), apresentando, considerando o intervalo de confiança, isoladamente os melhores valores do parâmetro comparado dentre todos os algoritmos para este cenário, com uma melhora variando de 36,9% a 46,0% comparado ao segundo melhor (`BALA`).

6.4.3 Análise de Resultados: Energia — Servidores

Mostramos na Figura 6.12 um gráfico da energia total consumida pelos servidores da nuvem. Neste cenário, avaliamos uma nuvem possuindo 100 servidores distribuídos em dois *data centers* geodistribuídos e homogêneos, empregando topologia *Binary Tree*. O recurso de migração de máquinas virtuais está ativado, com SLA de garantia de 100% de processamento para as máquinas virtuais, com comportamento de usuários indefinido e cargas de trabalho com múltiplas prioridades. O algoritmo de escalonamento que apre-

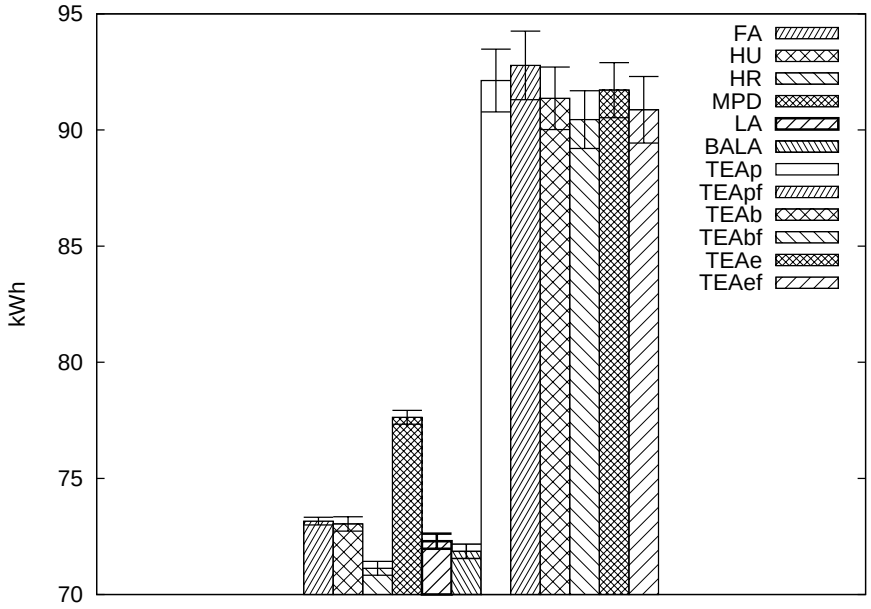

Figura 6.12: Energia Servidores: DC Geo. Homo., 100 srvs, Binary Tree, Mig, SLA, Comp Usr Ind, Mult Pr

sentou melhores resultados (71,13 ± 0,30 kWh) para este quesito foi o `HR`, apresentando, considerando o intervalo de confiança, isoladamente os melhores valores do parâmetro comparado dentre todos os algoritmos para este cenário, com uma melhora variando de 0,2% a 1,9% comparado ao segundo melhor (`BALA`).

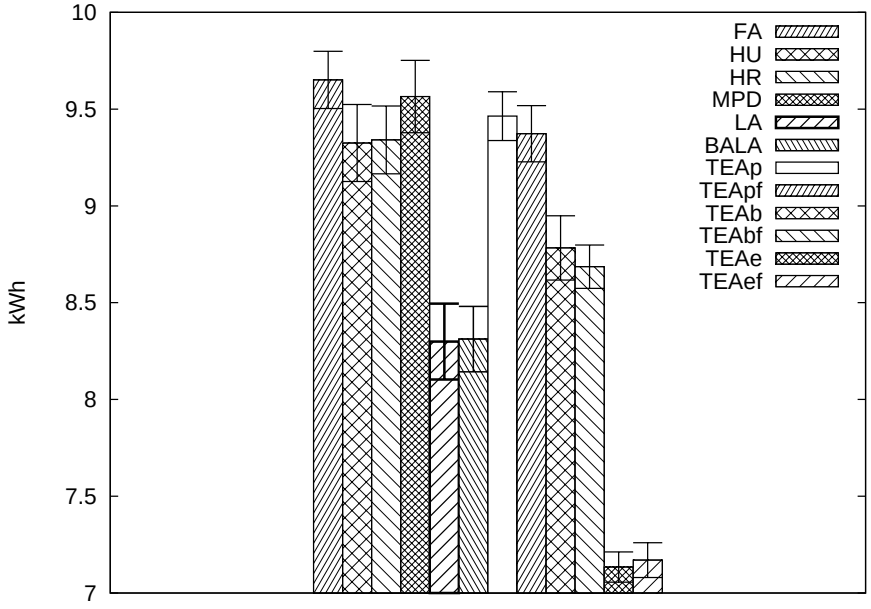

Figura 6.13: Energia Servidores: DC Geo. Het., 10 srvs, Flat-Tree, Mig, SLA, Comp Usr Ind, S/Mult Pr

A Figura 6.13 traz informações sobre a energia total consumida pelos servidores da nuvem. Neste cenário, avaliamos uma nuvem possuindo 10 servidores distribuídos em

dois *data centers* geodistribuídos e heterogêneos, empregando topologia *Flat-Tree*. O recurso de migração de máquinas virtuais está ativado, com SLA de garantia de 100% de processamento para as máquinas virtuais, com comportamento de usuários indefinido e cargas de trabalho com somente uma prioridade. O algoritmo de escalonamento que apresentou melhores resultados (7,13 ± 0,08 kWh) para este quesito foi o `TEA` (`TEAe`), apresentando, considerando o intervalo de confiança, isoladamente os melhores valores do parâmetro comparado dentre todos os algoritmos para este cenário, com uma melhora variando de 12,4% a 20,4% comparado ao segundo melhor (`LA`).

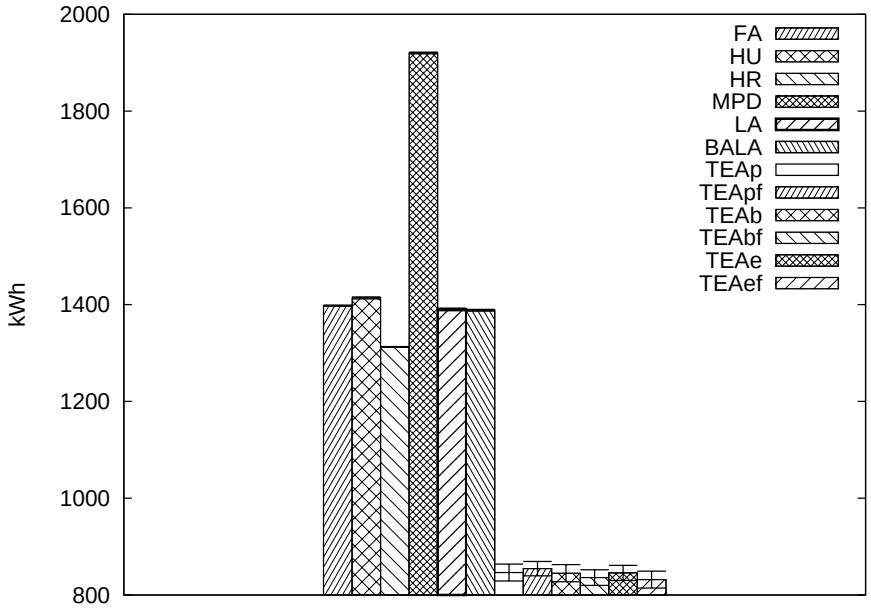

Figura 6.14: Energia Servidores: DC Geo. Homo., 1.000 srvs, Single-Hop Star, Mig, SLA, Comp Usr Ind, S/Mult Pr

Mostramos na Figura 6.14 um gráfico da energia total consumida pelos servidores da nuvem. Neste cenário, avaliamos uma nuvem possuindo 1.000 servidores distribuídos em dois *data centers* geodistribuídos e homogêneos, empregando topologia *Single-Hop Star*. O recurso de migração de máquinas virtuais está ativado, com SLA de garantia de 100% de processamento para as máquinas virtuais, com comportamento de usuários indefinido e cargas de trabalho com somente uma prioridade. O algoritmo de escalonamento que apresentou melhores resultados (831,78 ± 17,33 kWh) para este quesito foi o `TEA` (`TEAef`), apresentando, considerando o intervalo de confiança, isoladamente os melhores valores do parâmetro comparado dentre todos os algoritmos para este cenário, com uma melhora variando de 54,5% a 61,3% comparado ao segundo melhor (`HR`).

Apresentamos na Figura 6.15 uma comparação da energia total consumida pelos servidores da nuvem. Neste cenário, avaliamos uma nuvem possuindo 10 servidores distribuídos em um *data center* não geodistribuído e homogêneo, empregando topologia *Binary Tree*. O recurso de migração de máquinas virtuais está ativado, sem SLA de garantia de 100% de processamento para as máquinas virtuais, com comportamento de usuários definido e cargas de trabalho com múltiplas prioridades. O algoritmo de escalonamento que apresentou melhores resultados (6,54 ± 0,05 kWh) para este quesito foi o `FA`, apresentando,

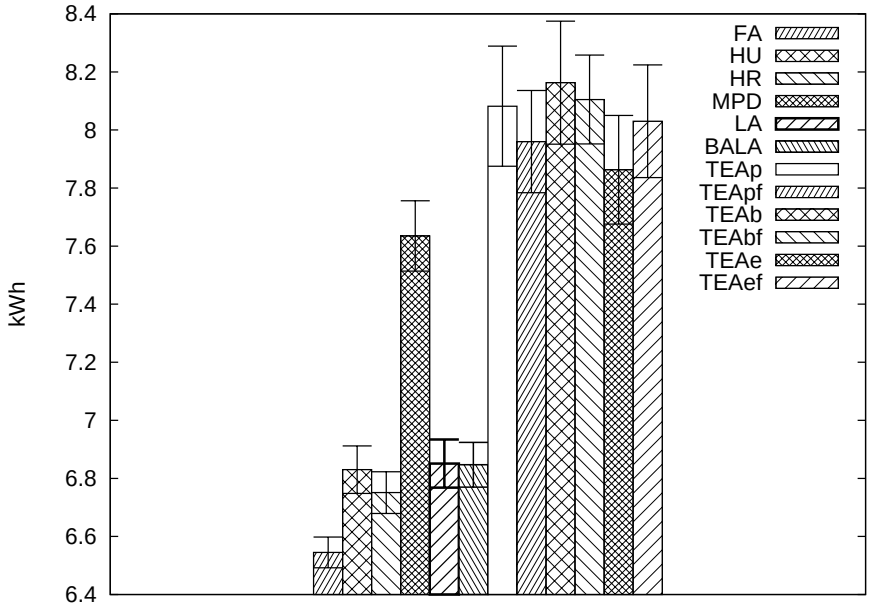

Figura 6.15: Energia Servidores: DC N.Geo. Homo., 10 srvs, Binary Tree, Mig, S/SLA, Comp Usr Def, Mult Pr

considerando o intervalo de confiança, isoladamente os melhores valores do parâmetro comparado dentre todos os algoritmos para este cenário, com uma melhora variando de 1,2% a 5,1% comparado ao segundo melhor (HR).

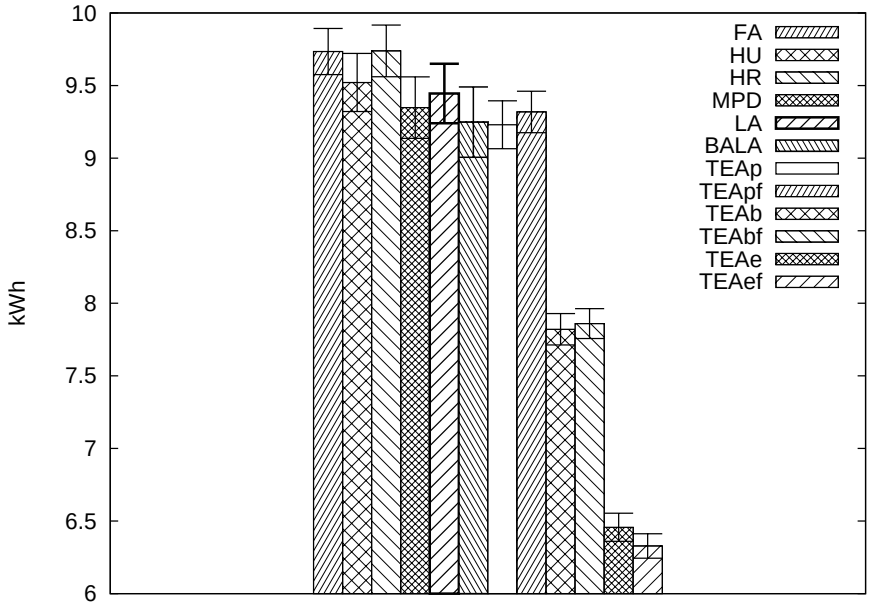

Figura 6.16: Energia Servidores: DC N.Geo. Het., 10 srvs, Binary Tree, S/Mig, SLA, Comp Usr Def, Mult Pr

Temos na Figura 6.16 um histograma sobre a energia total consumida pelos servidores da nuvem. Neste cenário, avaliamos uma nuvem possuindo 10 servidores distribuídos em um *data center* não geodistribuído e heterogêneo, empregando topologia *Binary Tree*.

O recurso de migração de máquinas virtuais está desativado, com SLA de garantia de 100% de processamento para as máquinas virtuais, com comportamento de usuários definido e cargas de trabalho com múltiplas prioridades. O algoritmo de escalonamento que apresentou melhores resultados (6,33 ± 0,08 kWh) para este quesito foi o `TEA` (`TEAef`), apresentando, considerando o intervalo de confiança, isoladamente os melhores valores do parâmetro comparado dentre todos os algoritmos para este cenário, com uma melhora variando de 40,5% a 52,0% comparado ao segundo melhor (`BALA`).

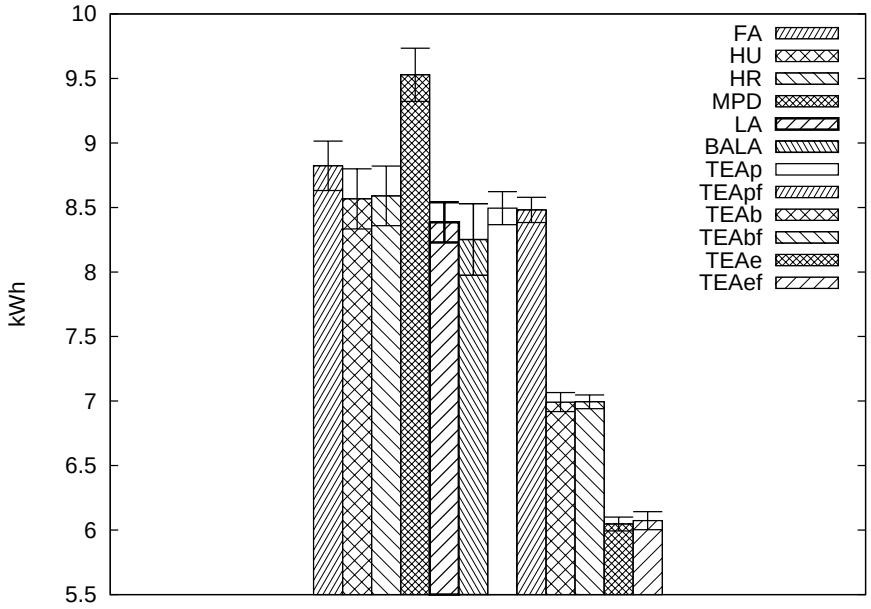

Figura 6.17: Energia Servidores: DC N.Geo. Het., 10 srvs, Flat-Tree, Mig, S/SLA, Comp Usr Ind, S/Mult Pr

Mostramos na Figura 6.17 um gráfico da energia total consumida pelos servidores da nuvem. Neste cenário, avaliamos uma nuvem possuindo 10 servidores distribuídos em um *data center* não geodistribuído e heterogêneo, empregando topologia *Flat-Tree*. O recurso de migração de máquinas virtuais está ativado, sem SLA de garantia de 100% de processamento para as máquinas virtuais, com comportamento de usuários indefinido e cargas de trabalho com somente uma prioridade. O algoritmo de escalonamento que apresentou melhores resultados (6,05 ± 0,05 kWh) para este quesito foi o `TEA` (`TEAe`), apresentando, considerando o intervalo de confiança, isoladamente os melhores valores do parâmetro comparado dentre todos os algoritmos para este cenário, com uma melhora variando de 30,7% a 42,3% comparado ao segundo melhor (`BALA`).

Temos na Figura 6.18 um histograma sobre a energia total consumida pelos servidores da nuvem. Neste cenário, avaliamos uma nuvem possuindo 10 servidores distribuídos em um *data center* não geodistribuído e heterogêneo, empregando topologia *Flat-Tree*. O recurso de migração de máquinas virtuais está desativado, com SLA de garantia de 100% de processamento para as máquinas virtuais, com comportamento de usuários definido e cargas de trabalho com múltiplas prioridades. O algoritmo de escalonamento que apresentou melhores resultados (6,30 ± 0,06 kWh) para este quesito foi o `TEA` (`TEAe`), apresentando, considerando o intervalo de confiança, isoladamente os melhores valores do

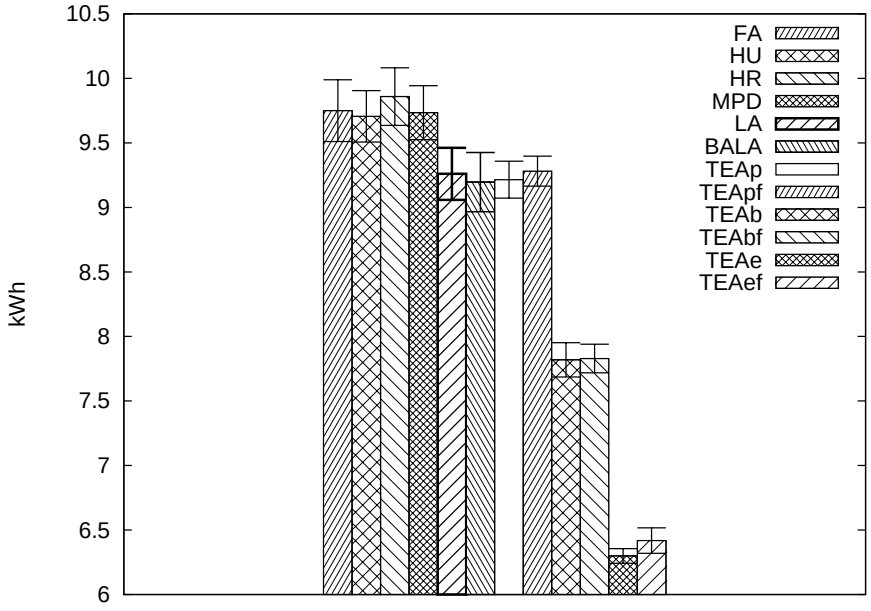

Figura 6.18: Energia Servidores: DC N.Geo. Het., 10 srvs, Flat-Tree, S/Mig, SLA, Comp Usr Def, Mult Pr

parâmetro comparado dentre todos os algoritmos para este cenário, com uma melhora variando de 41,0% a 51,0% comparado ao segundo melhor (`BALA`).

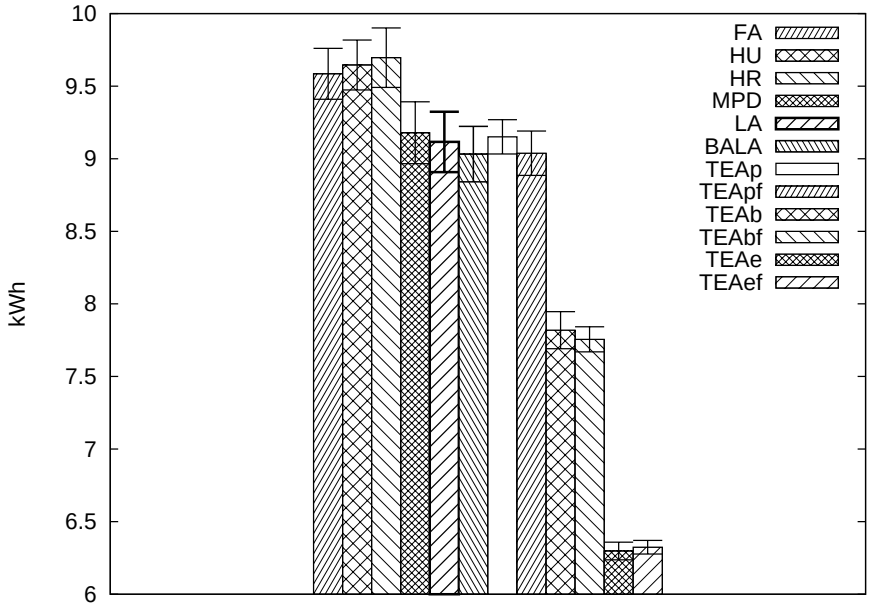

Figura 6.19: Energia Servidores: DC N.Geo. Het., 10 srvs, K-Ary Fat Tree, S/Mig, SLA, Comp Usr Ind, Mult Pr

Mostramos na Figura 6.19 um gráfico da energia total consumida pelos servidores da nuvem. Neste cenário, avaliamos uma nuvem possuindo 10 servidores distribuídos em um *data center* não geodistribuído e heterogêneo, empregando topologia *K-Ary Fat Tree*. O recurso de migração de máquinas virtuais está desativado, com SLA de garantia

de 100% de processamento para as máquinas virtuais, com comportamento de usuários indefinido e cargas de trabalho com múltiplas prioridades. O algoritmo de escalonamento que apresentou melhores resultados (6,30 ± 0,06 kWh) para este quesito foi o `TEA` (`TEAe`), apresentando, considerando o intervalo de confiança, isoladamente os melhores valores do parâmetro comparado dentre todos os algoritmos para este cenário, com uma melhora variando de 39,0% a 47,9% comparado ao segundo melhor (`BALA`).

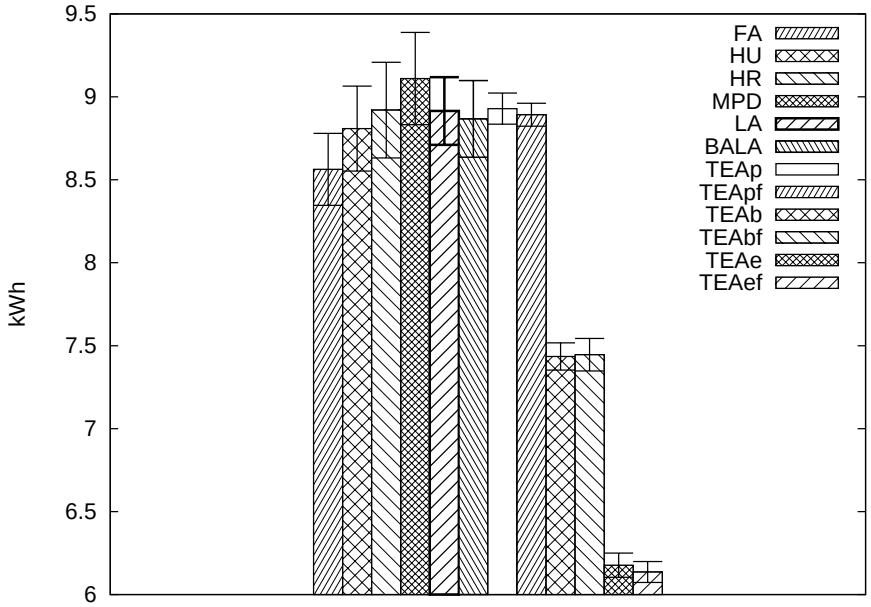

Figura 6.20: Energia Servidores: DC N.Geo. Het., 10 srvs, Single-Hop Star, S/Mig, S/SLA, Comp Usr Ind, S/Mult Pr

A Figura 6.20 traz informações sobre a energia total consumida pelos servidores da nuvem. Neste cenário, avaliamos uma nuvem possuindo 10 servidores distribuídos em um *data center* não geodistribuído e heterogêneo, empregando topologia *Single-Hop Star*. O recurso de migração de máquinas virtuais está desativado, sem SLA de garantia de 100% de processamento para as máquinas virtuais, com comportamento de usuários indefinido e cargas de trabalho com somente uma prioridade. O algoritmo de escalonamento que apresentou melhores resultados (6,14 ± 0,06 kWh) para este quesito foi o `TEA` (`TEAef`), apresentando, considerando o intervalo de confiança, isoladamente os melhores valores do parâmetro comparado dentre todos os algoritmos para este cenário, com uma melhora variando de 34,6% a 44,6% comparado ao segundo melhor (`FA`).

6.4.4 Análise de Resultados: Energia — Switches

Temos na Figura 6.21 um histograma sobre a energia total consumida pelos *switches* dos *data centers* da nuvem. Neste cenário, avaliamos uma nuvem possuindo 100 servidores distribuídos em dois *data centers* geodistribuídos e homogêneos, empregando topologia *Flat-Tree*. O recurso de migração de máquinas virtuais está ativado, com SLA de garantia de 100% de processamento para as máquinas virtuais, com comportamento de usuários definido e cargas de trabalho com múltiplas prioridades. O algoritmo de escalonamento

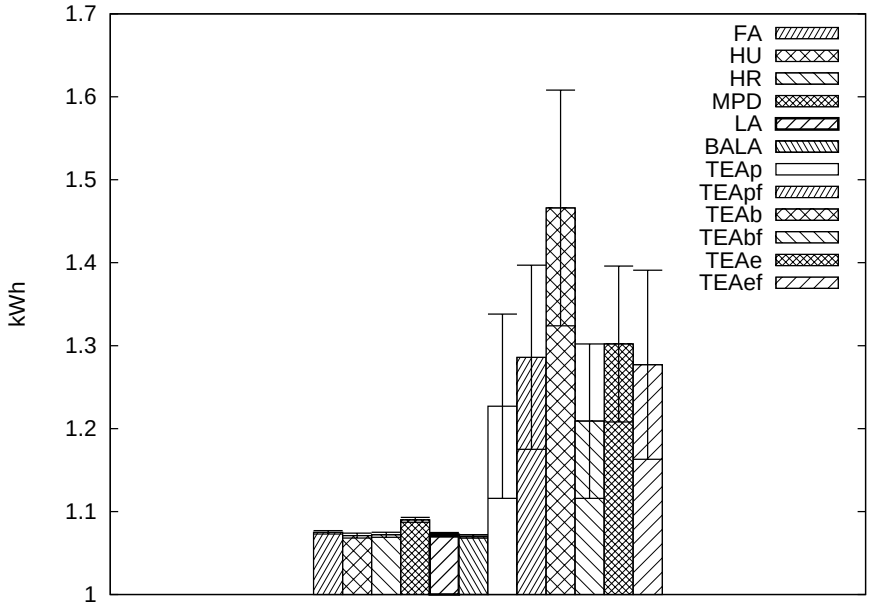

Figura 6.21: Energia Switches: DC Geo. Homo., 100 srvs, Flat-Tree, Mig, SLA, Comp Usr Def, Mult Pr

que apresentou melhores resultados (1,07 ± 0,00 kWh) para este quesito foi o `BALA`, empatado tecnicamente pelo menos com o segundo melhor, `HU` (1,07 ± 0,00 kWh).

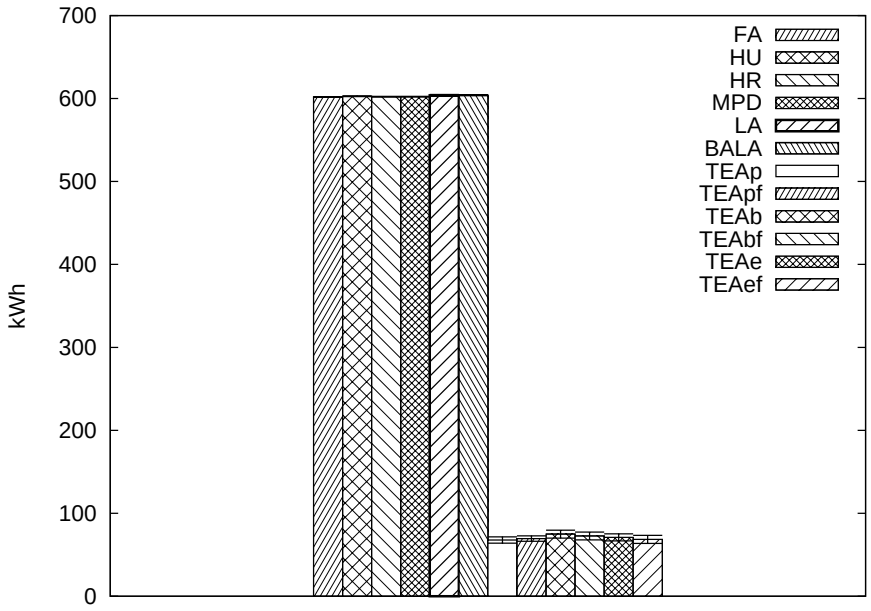

Figura 6.22: Energia Switches: DC Geo. Homo., 1.000 srvs, Flat-Tree, S/Mig, SLA, Comp Usr Def, S/Mult Pr

Mostramos na Figura 6.22 um gráfico da energia total consumida pelos *switches* dos *data centers* da nuvem. Neste cenário, avaliamos uma nuvem possuindo 1.000 servidores distribuídos em dois *data centers* geodistribuídos e homogêneos, empregando topologia *Flat-Tree*. O recurso de migração de máquinas virtuais está desativado, com SLA de

garantia de 100% de processamento para as máquinas virtuais, com comportamento de usuários definido e cargas de trabalho com somente uma prioridade. O algoritmo de escalonamento que apresentou melhores resultados (67,77 ± 3,74 kWh) para este quesito foi o `TEA` (`TEAp`), apresentando, considerando o intervalo de confiança, isoladamente os melhores valores do parâmetro comparado dentre todos os algoritmos para este cenário, com uma melhora variando de 741,1% a 840,5% comparado ao segundo melhor (`FA`).

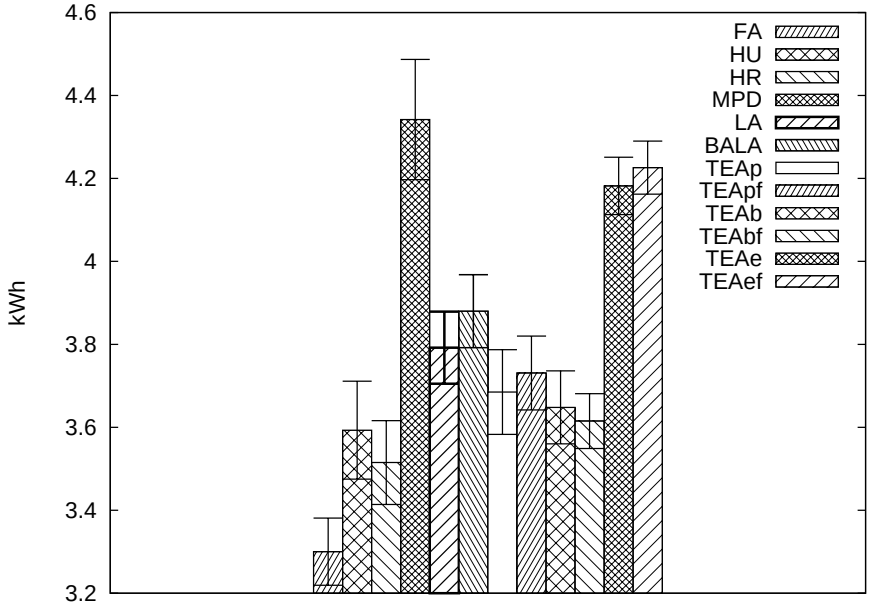

Figura 6.23: Energia Switches: DC N.Geo. Het., 10 srvs, Binary Tree, Mig, SLA, Comp Usr Def, S/Mult Pr

Temos na Figura 6.23 um histograma sobre a energia total consumida pelos *switches* dos *data centers* da nuvem. Neste cenário, avaliamos uma nuvem possuindo 10 servidores distribuídos em um *data center* não geodistribuído e heterogêneo, empregando topologia *Binary Tree*. O recurso de migração de máquinas virtuais está ativado, com SLA de garantia de 100% de processamento para as máquinas virtuais, com comportamento de usuários definido e cargas de trabalho com somente uma prioridade. O algoritmo de escalonamento que apresentou melhores resultados (3,30 ± 0,08 kWh) para este quesito foi o `FA`, apresentando, considerando o intervalo de confiança, isoladamente os melhores valores do parâmetro comparado dentre todos os algoritmos para este cenário, com uma melhora variando de 1,0% a 12,3% comparado ao segundo melhor (`HR`).

Temos na Figura 6.24 um histograma sobre a energia total consumida pelos *switches* dos *data centers* da nuvem. Neste cenário, avaliamos uma nuvem possuindo 10 servidores distribuídos em um *data center* não geodistribuído e heterogêneo, empregando topologia *Binary Tree*. O recurso de migração de máquinas virtuais está desativado, com SLA de garantia de 100% de processamento para as máquinas virtuais, com comportamento de usuários indefinido e cargas de trabalho com múltiplas prioridades. O algoritmo de escalonamento que apresentou melhores resultados (3,08 ± 0,07 kWh) para este quesito foi o `TEA` (`TEAbf`), empatado tecnicamente pelo menos com o segundo melhor, `HR` (3,18 ± 0,08 kWh).

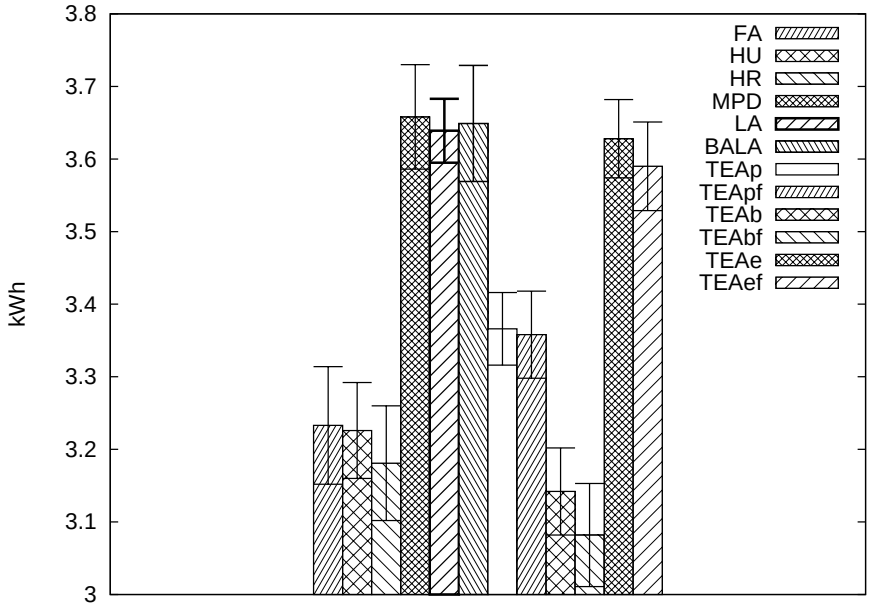

Figura 6.24: Energia Switches: DC N.Geo. Het., 10 srvs, Binary Tree, S/Mig, SLA, Comp Usr Ind, Mult Pr

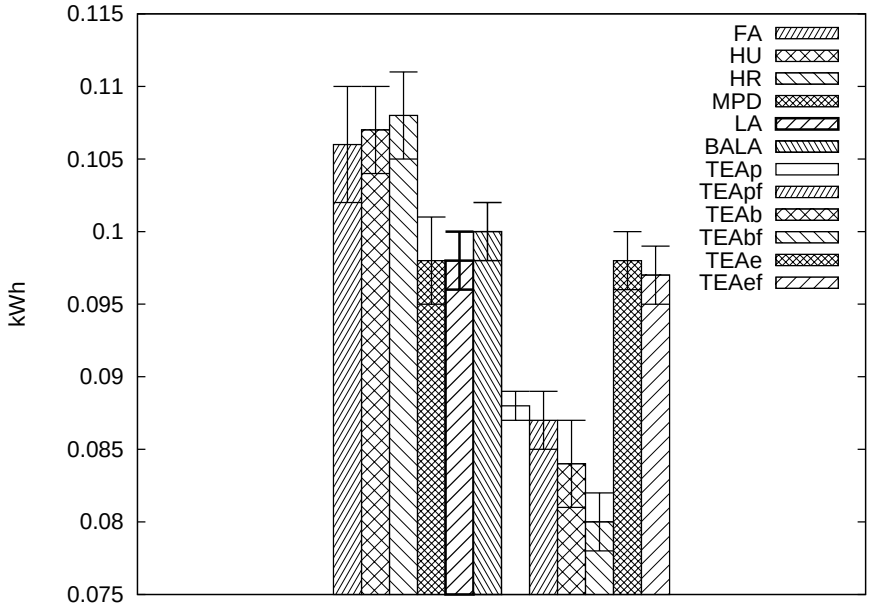

Figura 6.25: Energia Switches: DC N.Geo. Het., 10 srvs, Flat-Tree, S/Mig, S/SLA, Comp Usr Ind, S/Mult Pr

Temos na Figura 6.25 um histograma sobre a energia total consumida pelos *switches* dos *data centers* da nuvem. Neste cenário, avaliamos uma nuvem possuindo 10 servidores distribuídos em um *data center* não geodistribuído e heterogêneo, empregando topologia *Flat-Tree*. O recurso de migração de máquinas virtuais está desativado, sem SLA de garantia de 100% de processamento para as máquinas virtuais, com comportamento de usuários indefinido e cargas de trabalho com somente uma prioridade. O algoritmo de escalonamento que apresentou melhores resultados (0,08 $\pm$ 0,00 kWh) para este quesito

foi o TEA (TEAbf), apresentando, considerando o intervalo de confiança, isoladamente os melhores valores do parâmetro comparado dentre todos os algoritmos para este cenário, com uma melhora variando de 16,8% a 28,3% comparado ao segundo melhor (LA).

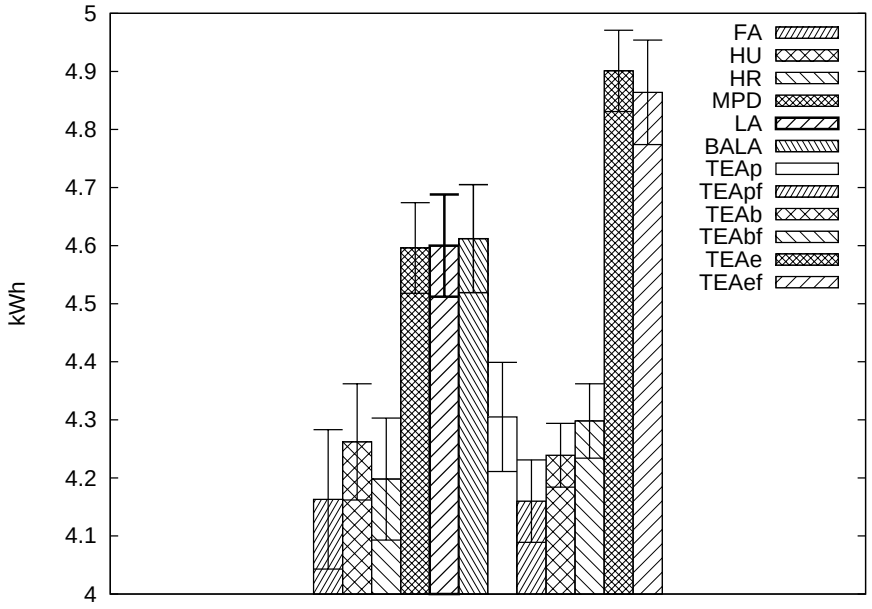

Figura 6.26: Energia Switches: DC N.Geo. Het., 10 srvs, K-Ary Fat Tree, S/Mig, SLA, Comp Usr Ind, S/Mult Pr

A Figura 6.26 traz informações sobre a energia total consumida pelos *switches* dos *data centers* da nuvem. Neste cenário, avaliamos uma nuvem possuindo 10 servidores distribuídos em um *data center* não geodistribuído e heterogêneo, empregando topologia *K-Ary Fat Tree*. O recurso de migração de máquinas virtuais está desativado, com SLA de garantia de 100% de processamento para as máquinas virtuais, com comportamento de usuários indefinido e cargas de trabalho com somente uma prioridade. O algoritmo de escalonamento que apresentou melhores resultados (4,16 ± 0,07 kWh) para este quesito foi o TEA (TEApf), empatado tecnicamente pelo menos com o segundo melhor, FA (4,16 ± 0,12 kWh).

A Figura 6.27 traz informações sobre a energia total consumida pelos *switches* dos *data centers* da nuvem. Neste cenário, avaliamos uma nuvem possuindo 10 servidores distribuídos em um *data center* não geodistribuído e heterogêneo, empregando topologia *Single-Hop Star*. O recurso de migração de máquinas virtuais está ativado, com SLA de garantia de 100% de processamento para as máquinas virtuais, com comportamento de usuários indefinido e cargas de trabalho com somente uma prioridade. O algoritmo de escalonamento que apresentou melhores resultados (0,10 ± 0,00 kWh) para este quesito foi o TEA (TEAb), apresentando, considerando o intervalo de confiança, isoladamente os melhores valores do parâmetro comparado dentre todos os algoritmos para este cenário, com uma melhora variando de 0,0% a 9,6% comparado ao segundo melhor (HR).

Mostramos na Figura 6.28 um gráfico da energia total consumida pelos *switches* dos *data centers* da nuvem. Neste cenário, avaliamos uma nuvem possuindo 10 servidores distribuídos em um *data center* não geodistribuído e heterogêneo, empregando topologia

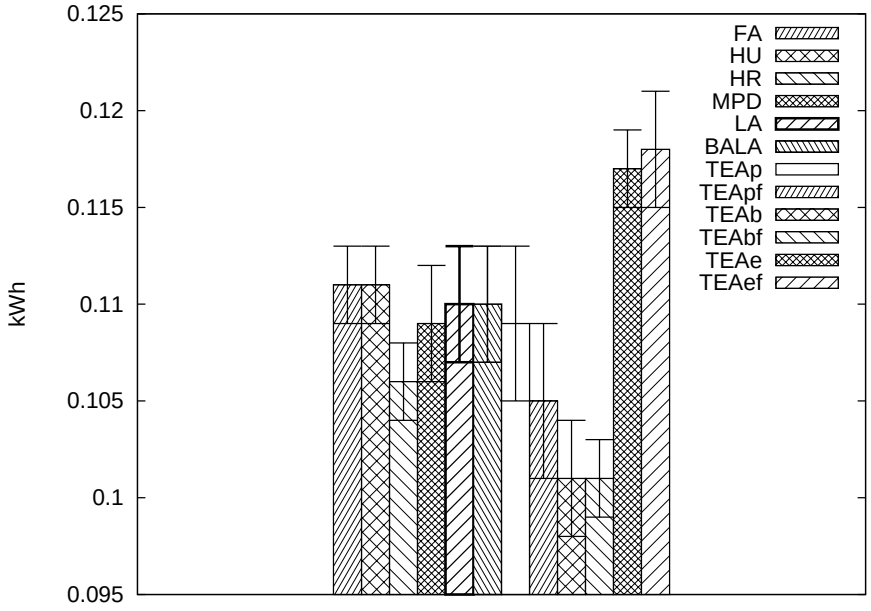

Figura 6.27: Energia Switches: DC N.Geo. Het., 10 srvs, Single-Hop Star, Mig, SLA, Comp Usr Ind, S/Mult Pr

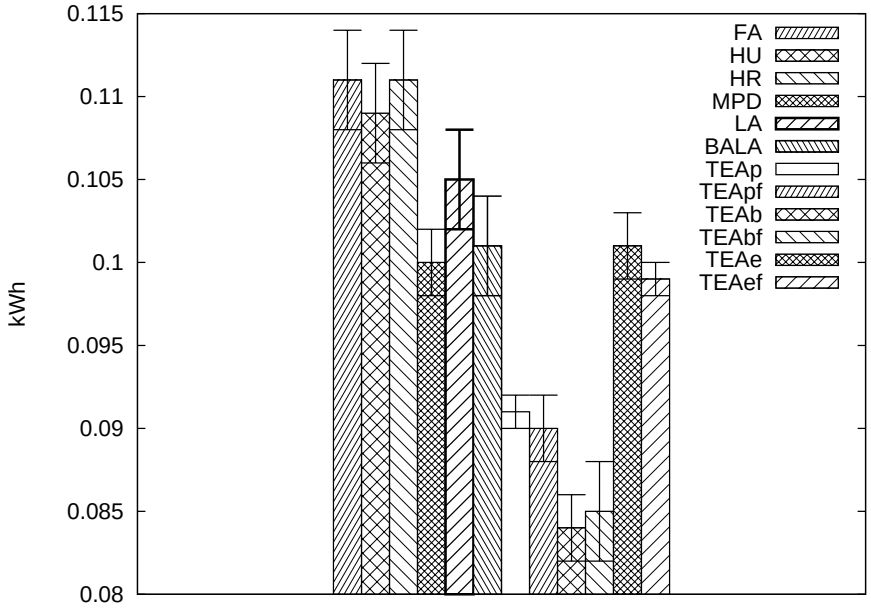

Figura 6.28: Energia Switches: DC N.Geo. Het., 10 srvs, Single-Hop Star, S/Mig, S/SLA, Comp Usr Ind, S/Mult Pr

Single-Hop Star. O recurso de migração de máquinas virtuais está desativado, sem SLA de garantia de 100% de processamento para as máquinas virtuais, com comportamento de usuários indefinido e cargas de trabalho com somente uma prioridade. O algoritmo de escalonamento que apresentou melhores resultados (0,08 ± 0,00 kWh) para este quesito foi o `TEA` (`TEAb`), apresentando, considerando o intervalo de confiança, isoladamente os melhores valores do parâmetro comparado dentre todos os algoritmos para este cenário, com uma melhora variando de 12,7% a 25,2% comparado ao segundo melhor (`MPD`).

6.4.5 Análise de Resultados: *Makespan*

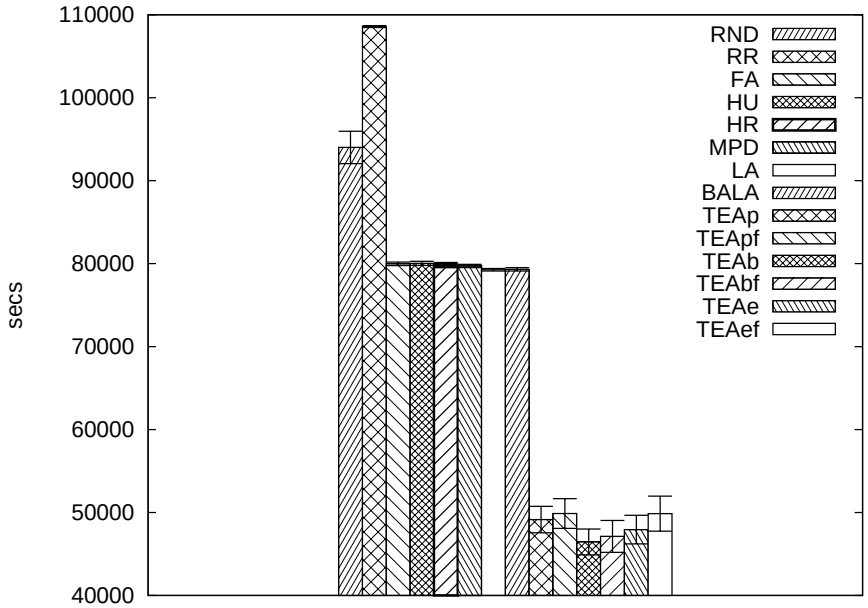

Figura 6.29: *Makespan*: DC Geo. Homo., 100 srvs, K-Ary Fat Tree, S/Mig, S/SLA, Comp Usr Def, Mult Pr

A Figura 6.29 traz informações sobre o *makespan*. Neste cenário, avaliamos uma nuvem possuindo 100 servidores distribuídos em dois *data centers* geodistribuídos e homogêneos, empregando topologia *K-Ary Fat Tree*. O recurso de migração de máquinas virtuais está desativado, sem SLA de garantia de 100% de processamento para as máquinas virtuais, com comportamento de usuários definido e cargas de trabalho com múltiplas prioridades. O algoritmo de escalonamento que apresentou melhores resultados (46.450,06 $\pm$ 1.563,15 s) para este quesito foi o `TEA` (`TEAb`), apresentando, considerando o intervalo de confiança, isoladamente os melhores valores do parâmetro comparado dentre todos os algoritmos para este cenário, com uma melhora variando de 64,8% a 76,9% comparado ao segundo melhor (`LA`).

Apresentamos na Figura 6.30 uma comparação do *makespan*. Neste cenário, avaliamos uma nuvem possuindo 1.000 servidores distribuídos em dois *data centers* geodistribuídos e homogêneos, empregando topologia *Single-Hop Star*. O recurso de migração de máquinas virtuais está ativado, com SLA de garantia de 100% de processamento para as máquinas virtuais, com comportamento de usuários indefinido e cargas de trabalho com somente uma prioridade. O algoritmo de escalonamento que apresentou melhores resultados (97.514,17 $\pm$ 5.278,83 s) para este quesito foi o `TEA` (`TEAef`), apresentando, considerando o intervalo de confiança, isoladamente os melhores valores do parâmetro comparado dentre todos os algoritmos para este cenário, com uma melhora variando de 569,9% a 646,7% comparado ao segundo melhor (`LA`).

Temos na Figura 6.31 um histograma sobre o *makespan*. Neste cenário, avaliamos uma nuvem possuindo 10 servidores distribuídos em dois *data centers* geodistribuídos e homogêneos, empregando topologia *Single-Hop Star*. O recurso de migração de máquinas

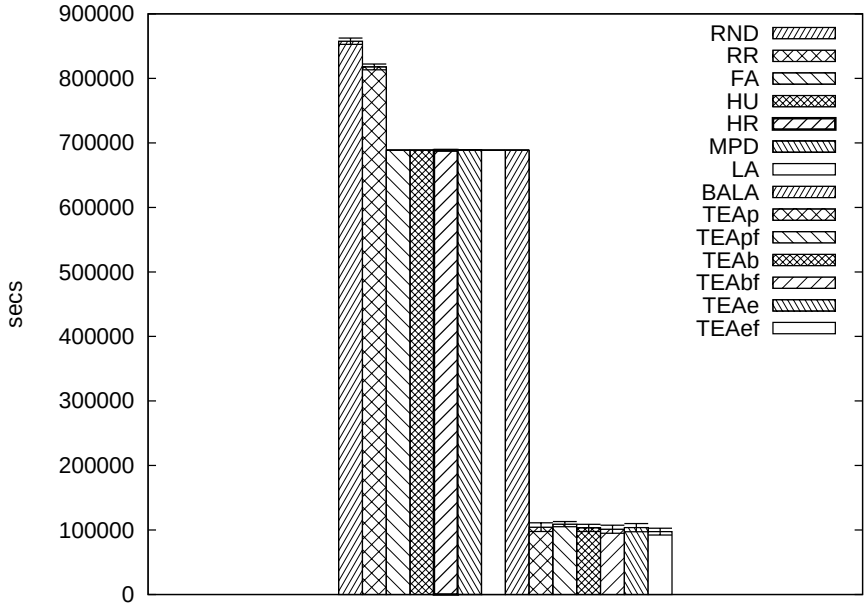

Figura 6.30: *Makespan*: DC Geo. Homo., 1.000 srvs, Single-Hop Star, Mig, SLA, Comp Usr Ind, S/Mult Pr

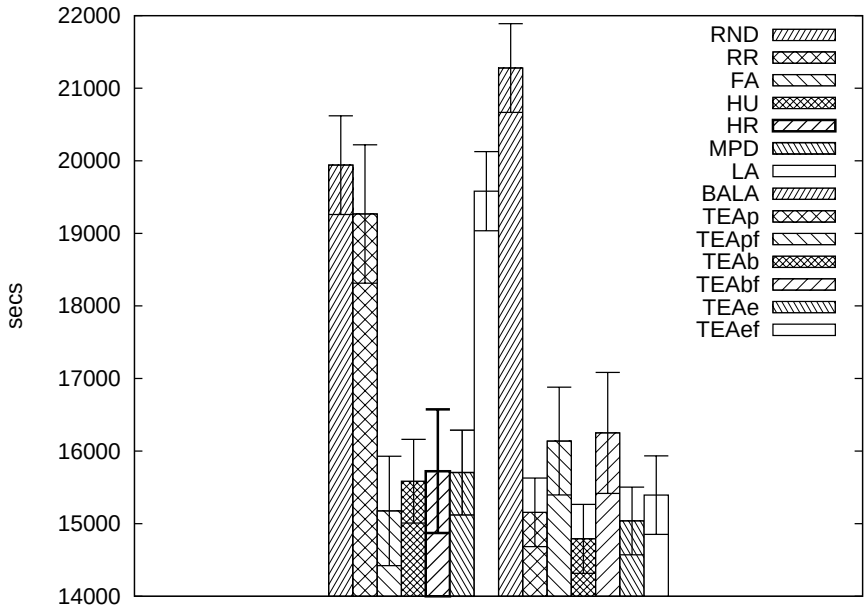

Figura 6.31: *Makespan*: DC Geo. Homo., 10 srvs, Single-Hop Star, Mig, SLA, Comp Usr Def, Mult Pr

virtuais está ativado, com SLA de garantia de 100% de processamento para as máquinas virtuais, com comportamento de usuários definido e cargas de trabalho com múltiplas prioridades. O algoritmo de escalonamento que apresentou melhores resultados (14.791,14 ± 474,75 s) para este quesito foi o `TEA` (`TEAb`), empatado tecnicamente pelo menos com o segundo melhor, `FA` (15.176,05 ± 753,81 s).

Mostramos na Figura 6.32 um gráfico do *makespan*. Neste cenário, avaliamos uma nuvem possuindo 10 servidores distribuídos em dois *data centers* geodistribuídos e he-

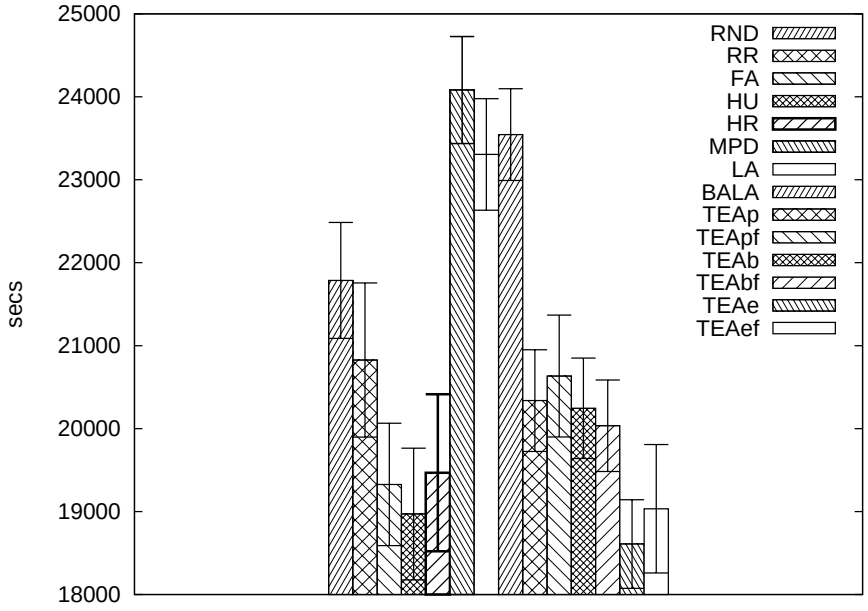

Figura 6.32: *Makespan*: DC Geo. Het., 10 srvs, Single-Hop Star, S/Mig, S/SLA, Comp Usr Def, S/Mult Pr

terogêneos, empregando topologia *Single-Hop Star*. O recurso de migração de máquinas virtuais está desativado, sem SLA de garantia de 100% de processamento para as máquinas virtuais, com comportamento de usuários definido e cargas de trabalho com somente uma prioridade. O algoritmo de escalonamento que apresentou melhores resultados (18.608,63 ± 534,11 s) para este quesito foi o `TEA` (`TEAe`), empatado tecnicamente pelo menos com o segundo melhor, `HU` (18.970,24 ± 793,31 s).

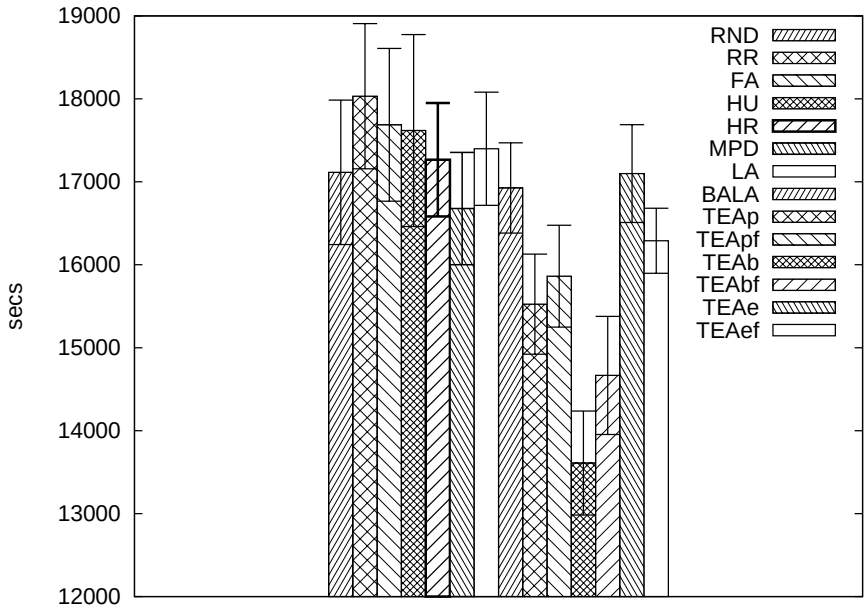

Figura 6.33: *Makespan*: DC N.Geo. Het., 10 srvs, Binary Tree, S/Mig, S/SLA, Comp Usr Def, S/Mult Pr

Mostramos na Figura 6.33 um gráfico do *makespan*. Neste cenário, avaliamos uma nuvem possuindo 10 servidores distribuídos em um *data center* não geodistribuído e heterogêneo, empregando topologia *Binary Tree*. O recurso de migração de máquinas virtuais está desativado, sem SLA de garantia de 100% de processamento para as máquinas virtuais, com comportamento de usuários definido e cargas de trabalho com somente uma prioridade. O algoritmo de escalonamento que apresentou melhores resultados (13.609,61 ± 626,55 s) para este quesito foi o `TEA` (`TEAb`), apresentando, considerando o intervalo de confiança, isoladamente os melhores valores do parâmetro comparado dentre todos os algoritmos para este cenário, com uma melhora variando de 12,4% a 33,7% comparado ao segundo melhor (`MPD`).

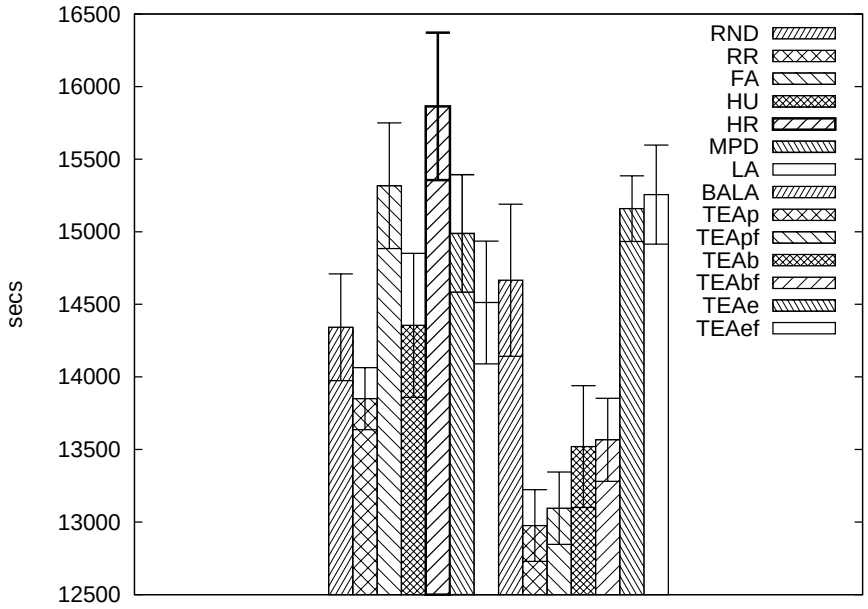

Figura 6.34: *Makespan*: DC N.Geo. Het., 10 srvs, Binary Tree, S/Mig, S/SLA, Comp Usr Ind, Mult Pr

A Figura 6.34 traz informações sobre o *makespan*. Neste cenário, avaliamos uma nuvem possuindo 10 servidores distribuídos em um *data center* não geodistribuído e heterogêneo, empregando topologia *Binary Tree*. O recurso de migração de máquinas virtuais está desativado, sem SLA de garantia de 100% de processamento para as máquinas virtuais, com comportamento de usuários indefinido e cargas de trabalho com múltiplas prioridades. O algoritmo de escalonamento que apresentou melhores resultados (12.976,20 ± 246,67 s) para este quesito foi o `TEA` (`TEAp`), apresentando, considerando o intervalo de confiança, isoladamente os melhores valores do parâmetro comparado dentre todos os algoritmos para este cenário, com uma melhora variando de 3,1% a 10,5% comparado ao segundo melhor (`RR`).

Apresentamos na Figura 6.35 uma comparação do *makespan*. Neste cenário, avaliamos uma nuvem possuindo 10 servidores distribuídos em um *data center* não geodistribuído e heterogêneo, empregando topologia *Binary Tree*. O recurso de migração de máquinas virtuais está desativado, sem SLA de garantia de 100% de processamento para as máquinas virtuais, com comportamento de usuários indefinido e cargas de trabalho com

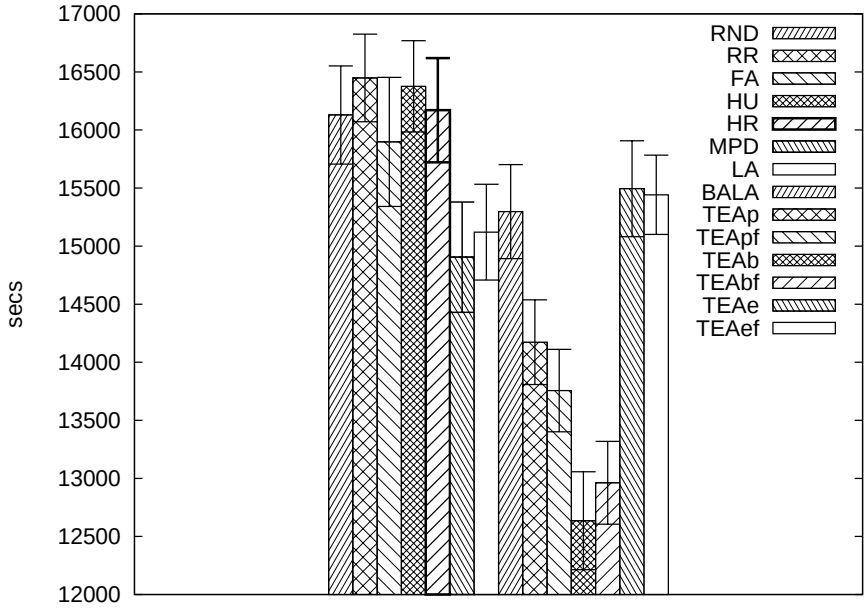

Figura 6.35: *Makespan*: DC N.Geo. Het., 10 srvs, Binary Tree, S/Mig, S/SLA, Comp Usr Ind, S/Mult Pr

somente uma prioridade. O algoritmo de escalonamento que apresentou melhores resultados (12.635,36 $\pm$ 421,94 s) para este quesito foi o `TEA` (`TEAb`), apresentando, considerando o intervalo de confiança, isoladamente os melhores valores do parâmetro comparado dentre todos os algoritmos para este cenário, com uma melhora variando de 10,5% a 25,9% comparado ao segundo melhor (`MPD`).

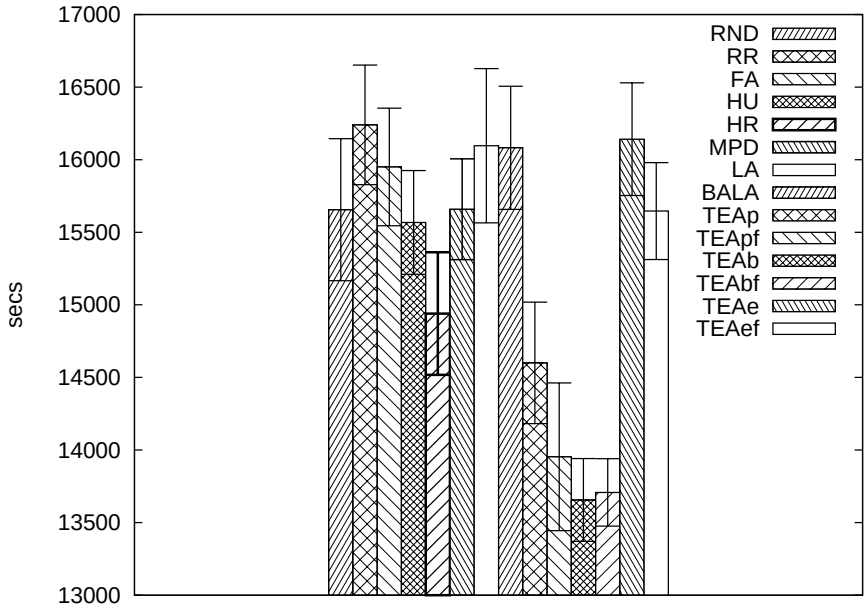

Figura 6.36: *Makespan*: DC N.Geo. Het., 10 srvs, Binary Tree, S/Mig, SLA, Comp Usr Ind, S/Mult Pr

Temos na Figura 6.36 um histograma sobre o *makespan*. Neste cenário, avaliamos uma nuvem possuindo 10 servidores distribuídos em um *data center* não geodistribuído e heterogêneo, empregando topologia *Binary Tree*. O recurso de migração de máquinas virtuais está desativado, com SLA de garantia de 100% de processamento para as máquinas virtuais, com comportamento de usuários indefinido e cargas de trabalho com somente uma prioridade. O algoritmo de escalonamento que apresentou melhores resultados (13.655,89 ± 285,06 s) para este quesito foi o `TEA` (`TEAb`), apresentando, considerando o intervalo de confiança, isoladamente os melhores valores do parâmetro comparado dentre todos os algoritmos para este cenário, com uma melhora variando de 4,1% a 14,9% comparado ao segundo melhor (`HR`).

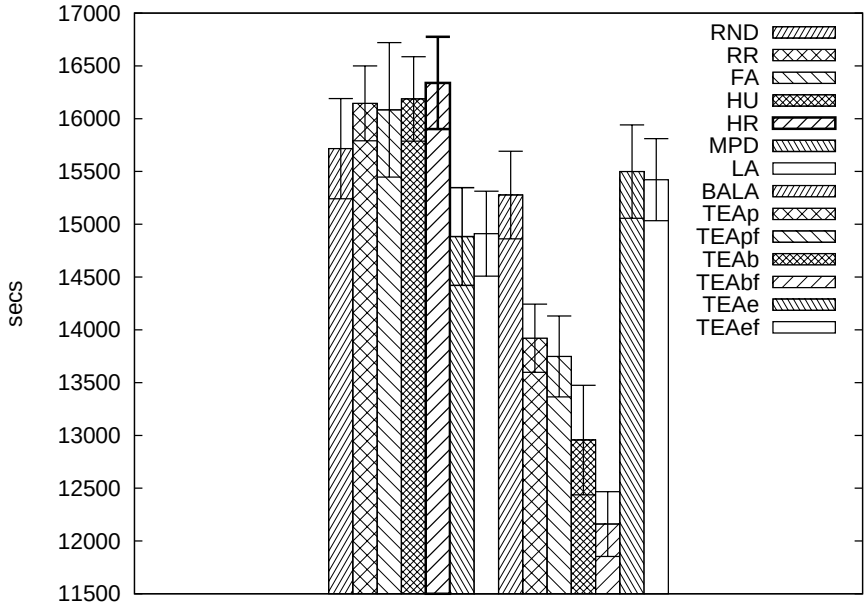

Figura 6.37: *Makespan*: DC N.Geo. Het., 10 srvs, Flat-Tree, S/Mig, S/SLA, Comp Usr Ind, S/Mult Pr

A Figura 6.37 traz informações sobre o *makespan*. Neste cenário, avaliamos uma nuvem possuindo 10 servidores distribuídos em um *data center* não geodistribuído e heterogêneo, empregando topologia *Flat-Tree*. O recurso de migração de máquinas virtuais está desativado, sem SLA de garantia de 100% de processamento para as máquinas virtuais, com comportamento de usuários indefinido e cargas de trabalho com somente uma prioridade. O algoritmo de escalonamento que apresentou melhores resultados (12.160,12 ± 307,30 s) para este quesito foi o `TEA` (`TEAbf`), apresentando, considerando o intervalo de confiança, isoladamente os melhores valores do parâmetro comparado dentre todos os algoritmos para este cenário, com uma melhora variando de 15,7% a 29,5% comparado ao segundo melhor (`MPD`).

Temos na Figura 6.38 um histograma sobre o *makespan*. Neste cenário, avaliamos uma nuvem possuindo 10 servidores distribuídos em um *data center* não geodistribuído e heterogêneo, empregando topologia *K-Ary Fat Tree*. O recurso de migração de máquinas virtuais está desativado, sem SLA de garantia de 100% de processamento para as máquinas virtuais, com comportamento de usuários indefinido e cargas de trabalho com

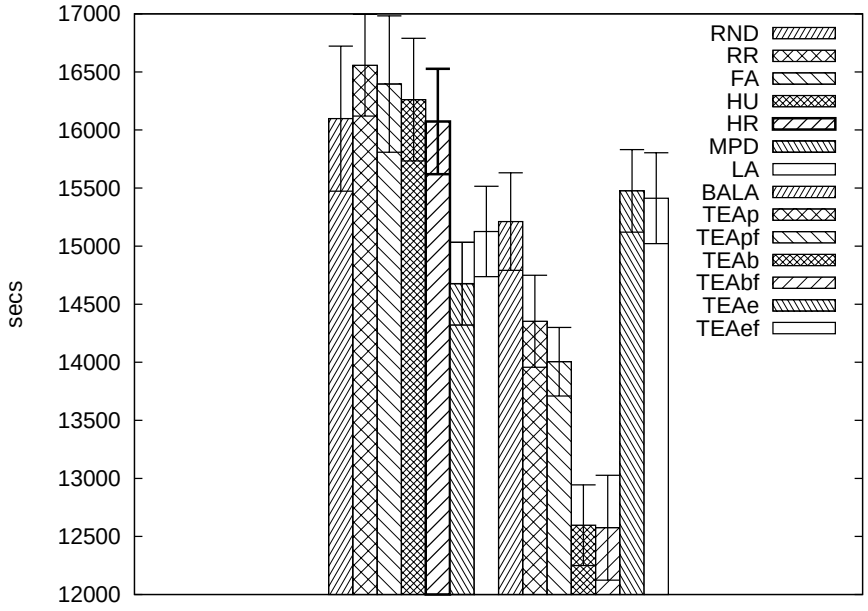

Figura 6.38: *Makespan*: DC N.Geo. Het., 10 srvs, K-Ary Fat Tree, S/Mig, S/SLA, Comp Usr Ind, S/Mult Pr

somente uma prioridade. O algoritmo de escalonamento que apresentou melhores resultados (12.575,15 $\pm$ 451,32 s) para este quesito foi o `TEA` (`TEAbf`), apresentando, considerando o intervalo de confiança, isoladamente os melhores valores do parâmetro comparado dentre todos os algoritmos para este cenário, com uma melhora variando de 9,9% a 24,0% comparado ao segundo melhor (`MPD`).

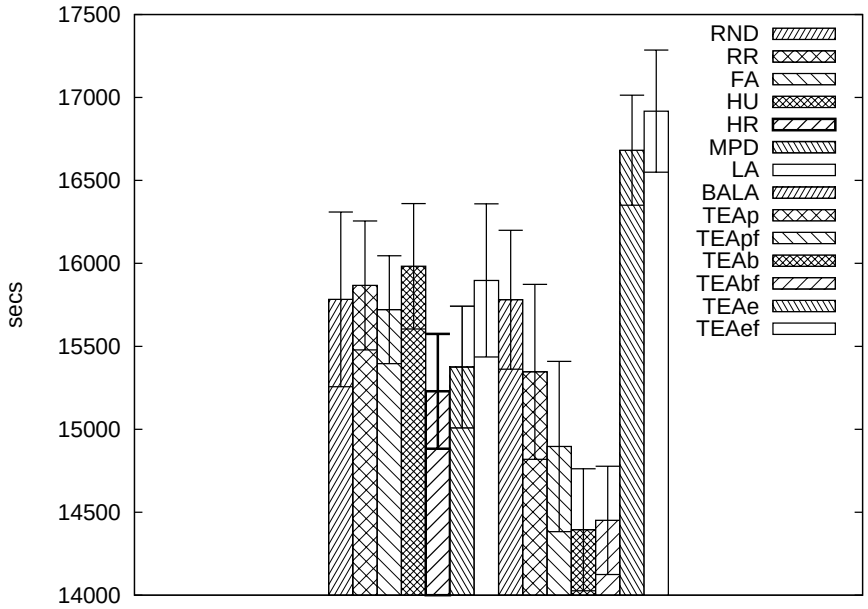

Figura 6.39: *Makespan*: DC N.Geo. Het., 10 srvs, Single-Hop Star, Mig, SLA, Comp Usr Ind, S/Mult Pr

Mostramos na Figura 6.39 um gráfico do *makespan*. Neste cenário, avaliamos uma nuvem possuindo 10 servidores distribuídos em um *data center* não geodistribuído e heterogêneo, empregando topologia *Single-Hop Star*. O recurso de migração de máquinas virtuais está ativado, com SLA de garantia de 100% de processamento para as máquinas virtuais, com comportamento de usuários indefinido e cargas de trabalho com somente uma prioridade. O algoritmo de escalonamento que apresentou melhores resultados (14.394,29 ± 367,40 s) para este quesito foi o `TEA` (`TEAb`), apresentando, considerando o intervalo de confiança, isoladamente os melhores valores do parâmetro comparado dentre todos os algoritmos para este cenário, com uma melhora variando de 0,8% a 11,0% comparado ao segundo melhor (`HR`).

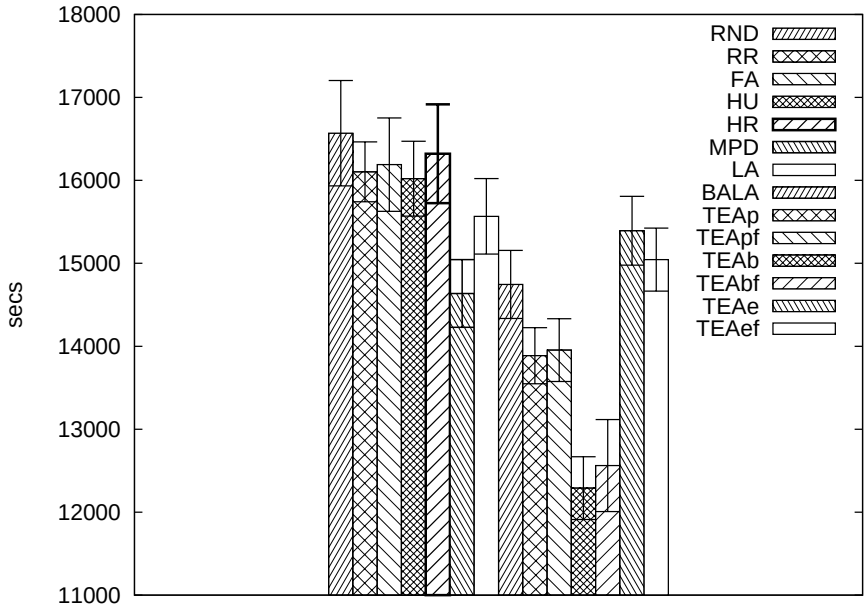

Figura 6.40: *Makespan*: DC N.Geo. Het., 10 srvs, Single-Hop Star, S/Mig, S/SLA, Comp Usr Ind, S/Mult Pr

A Figura 6.40 traz informações sobre o *makespan*. Neste cenário, avaliamos uma nuvem possuindo 10 servidores distribuídos em um *data center* não geodistribuído e heterogêneo, empregando topologia *Single-Hop Star*. O recurso de migração de máquinas virtuais está desativado, sem SLA de garantia de 100% de processamento para as máquinas virtuais, com comportamento de usuários indefinido e cargas de trabalho com somente uma prioridade. O algoritmo de escalonamento que apresentou melhores resultados (12.289,87 ± 378,03 s) para este quesito foi o `TEA` (`TEAb`), apresentando, considerando o intervalo de confiança, isoladamente os melhores valores do parâmetro comparado dentre todos os algoritmos para este cenário, com uma melhora variando de 12,3% a 26,3% comparado ao segundo melhor (`MPD`).

6.4.6 Análise de Resultados: Tempo de Processamento de Cargas

Mostramos na Figura 6.41 um gráfico do tempo de processamento das cargas de trabalho após a instanciação da máquina virtual. Neste cenário, avaliamos uma nuvem

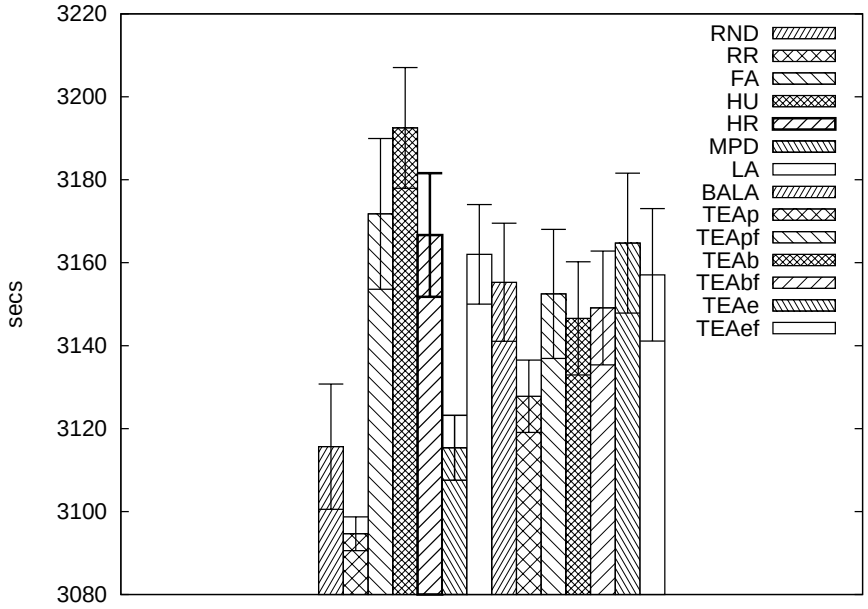

Figura 6.41: Tempo de Processamento de Cargas: DC Geo. Het., 10 srvs, Binary Tree, S/Mig, S/SLA, Comp Usr Def, Mult Pr

possuindo 10 servidores distribuídos em dois *data centers* geodistribuídos e heterogêneos, empregando topologia *Binary Tree*. O recurso de migração de máquinas virtuais está desativado, sem SLA de garantia de 100% de processamento para as máquinas virtuais, com comportamento de usuários definido e cargas de trabalho com múltiplas prioridades. O algoritmo de escalonamento que apresentou melhores resultados (3.094,64 ± 4,07 s) para este quesito foi o `RR`, apresentando, considerando o intervalo de confiança, isoladamente os melhores valores do parâmetro comparado dentre todos os algoritmos para este cenário, com uma melhora variando de 0,3% a 1,1% comparado ao segundo melhor (`MPD`).

Mostramos na Figura 6.42 um gráfico do tempo de processamento das cargas de trabalho após a instanciação da máquina virtual. Neste cenário, avaliamos uma nuvem possuindo 10 servidores distribuídos em dois *data centers* geodistribuídos e homogêneos, empregando topologia *Binary Tree*. O recurso de migração de máquinas virtuais está desativado, sem SLA de garantia de 100% de processamento para as máquinas virtuais, com comportamento de usuários definido e cargas de trabalho com somente uma prioridade. O algoritmo de escalonamento que apresentou melhores resultados (3.087,03 ± 4,11 s) para este quesito foi o `BALA`, empatado tecnicamente pelo menos com o segundo melhor, `RR` (3.089,30 ± 4,46 s).

Temos na Figura 6.43 um histograma sobre o tempo de processamento das cargas de trabalho após a instanciação da máquina virtual. Neste cenário, avaliamos uma nuvem possuindo 10 servidores distribuídos em dois *data centers* geodistribuídos e homogêneos, empregando topologia *Flat-Tree*. O recurso de migração de máquinas virtuais está ativado, com SLA de garantia de 100% de processamento para as máquinas virtuais, com comportamento de usuários definido e cargas de trabalho com somente uma prioridade. O algoritmo de escalonamento que apresentou melhores resultados (3.101,26 ± 4,04 s) para

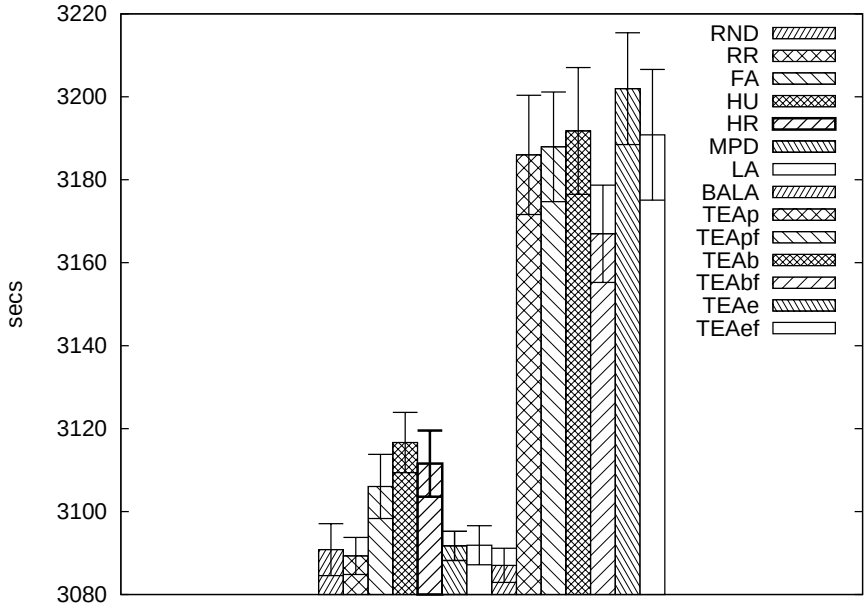

Figura 6.42: Tempo de Processamento de Cargas: DC Geo. Homo., 10 srvs, Binary Tree, S/Mig, S/SLA, Comp Usr Def, S/Mult Pr

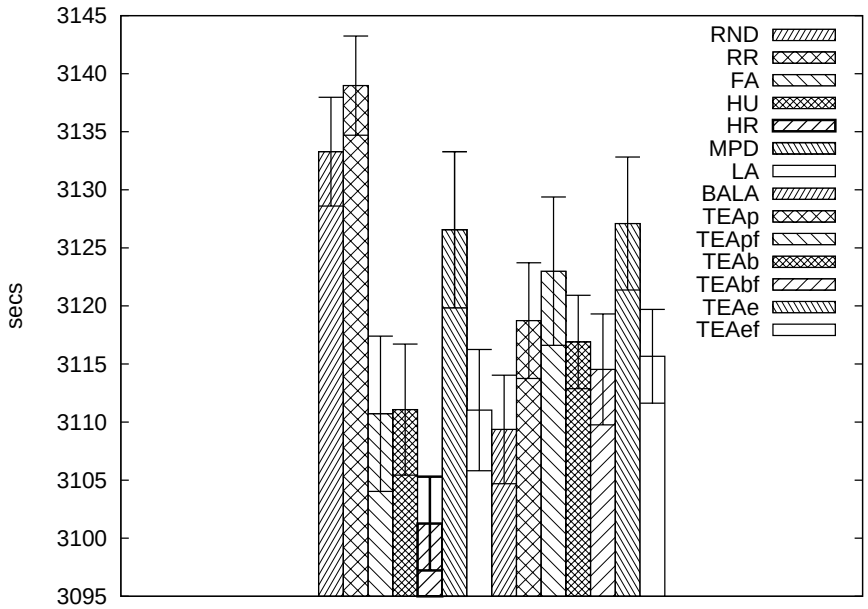

Figura 6.43: Tempo de Processamento de Cargas: DC Geo. Homo., 10 srvs, Flat-Tree, Mig, SLA, Comp Usr Def, S/Mult Pr

este quesito foi o `HR`, empatado tecnicamente pelo menos com o segundo melhor, `BALA` (3.109,37 $\pm$ 4,67 s).

Mostramos na Figura 6.44 um gráfico do tempo de processamento das cargas de trabalho após a instanciação da máquina virtual. Neste cenário, avaliamos uma nuvem possuindo 100 servidores distribuídos em dois *data centers* geodistribuídos e heterogêneos, empregando topologia *Single-Hop Star*. O recurso de migração de máquinas virtuais está ativado, sem SLA de garantia de 100% de processamento para as máquinas virtuais, com

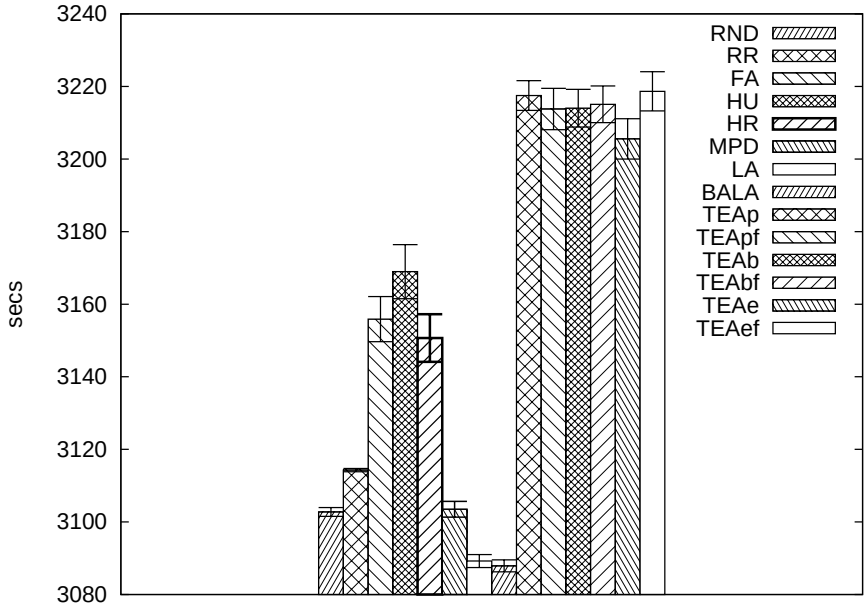

Figura 6.44: Tempo de Processamento de Cargas: DC Geo. Het., 100 srvs, Single-Hop Star, Mig, S/SLA, Comp Usr Def, S/Mult Pr

comportamento de usuários definido e cargas de trabalho com somente uma prioridade. O algoritmo de escalonamento que apresentou melhores resultados (3.087,90 ± 1,66 s) para este quesito foi o `BALA`, empatado tecnicamente pelo menos com o segundo melhor, `LA` (3.089,23 ± 1,78 s).

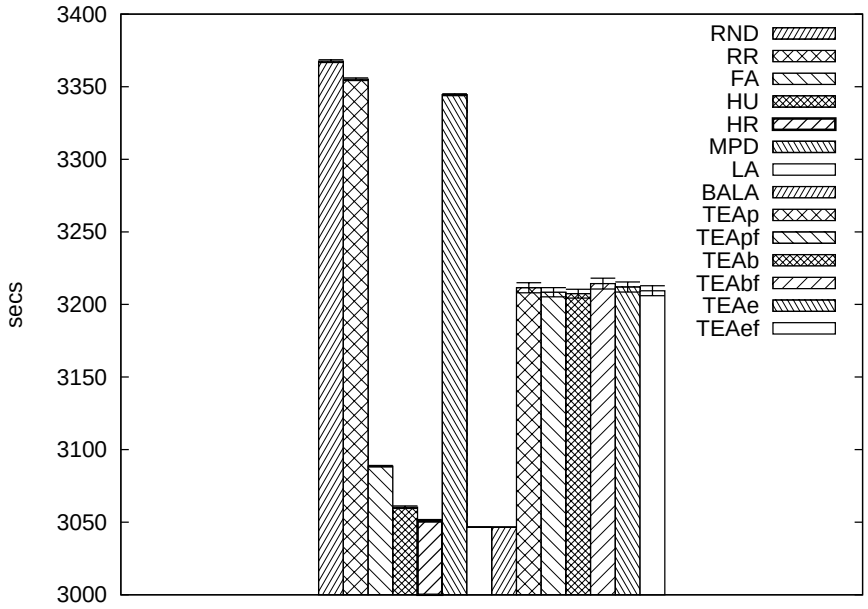

Figura 6.45: Tempo de Processamento de Cargas: DC Geo. Homo., 1.000 srvs, Single-Hop Star, Mig, S/SLA, Comp Usr Def, S/Mult Pr

Mostramos na Figura 6.45 um gráfico do tempo de processamento das cargas de trabalho após a instanciação da máquina virtual. Neste cenário, avaliamos uma nuvem pos-

suindo 1.000 servidores distribuídos em dois *data centers* geodistribuídos e homogêneos, empregando topologia *Single-Hop Star*. O recurso de migração de máquinas virtuais está ativado, sem SLA de garantia de 100% de processamento para as máquinas virtuais, com comportamento de usuários definido e cargas de trabalho com somente uma prioridade. O algoritmo de escalonamento que apresentou melhores resultados (3.046,56 ± 0,12 s) para este quesito foi o `LA`, empatado tecnicamente pelo menos com o segundo melhor, `BALA` (3.046,58 ± 0,09 s).

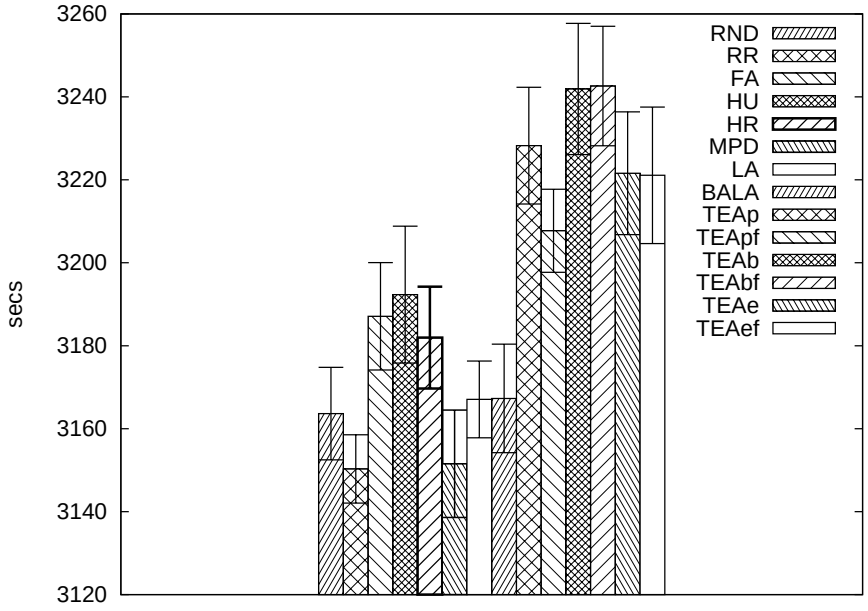

Figura 6.46: Tempo de Processamento de Cargas: DC N.Geo. Het., 10 srvs, Binary Tree, Mig, S/SLA, Comp Usr Def, Mult Pr

Apresentamos na Figura 6.46 uma comparação do tempo de processamento das cargas de trabalho após a instanciação da máquina virtual. Neste cenário, avaliamos uma nuvem possuindo 10 servidores distribuídos em um *data center* não geodistribuído e heterogêneo, empregando topologia *Binary Tree*. O recurso de migração de máquinas virtuais está ativado, sem SLA de garantia de 100% de processamento para as máquinas virtuais, com comportamento de usuários definido e cargas de trabalho com múltiplas prioridades. O algoritmo de escalonamento que apresentou melhores resultados (3.150,29 ± 8,23 s) para este quesito foi o `RR`, empatado tecnicamente pelo menos com o segundo melhor, `MPD` (3.151,55 ± 12,95 s).

A Figura 6.47 traz informações sobre o tempo de processamento das cargas de trabalho após a instanciação da máquina virtual. Neste cenário, avaliamos uma nuvem possuindo 10 servidores distribuídos em um *data center* não geodistribuído e heterogêneo, empregando topologia *Flat-Tree*. O recurso de migração de máquinas virtuais está desativado, sem SLA de garantia de 100% de processamento para as máquinas virtuais, com comportamento de usuários indefinido e cargas de trabalho com somente uma prioridade. O algoritmo de escalonamento que apresentou melhores resultados (3.083,46 ± 2,75 s) para este quesito foi o `RR`, empatado tecnicamente pelo menos com o segundo melhor, `MPD` (3.094,99 ± 11,12 s).

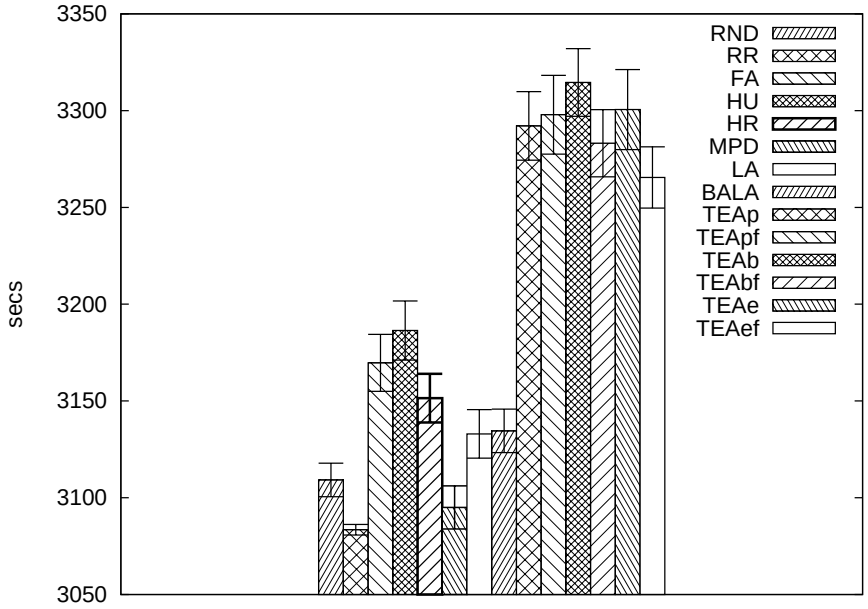

Figura 6.47: Tempo de Processamento de Cargas: DC N.Geo. Het., 10 srvs, Flat-Tree, S/Mig, S/SLA, Comp Usr Ind, S/Mult Pr

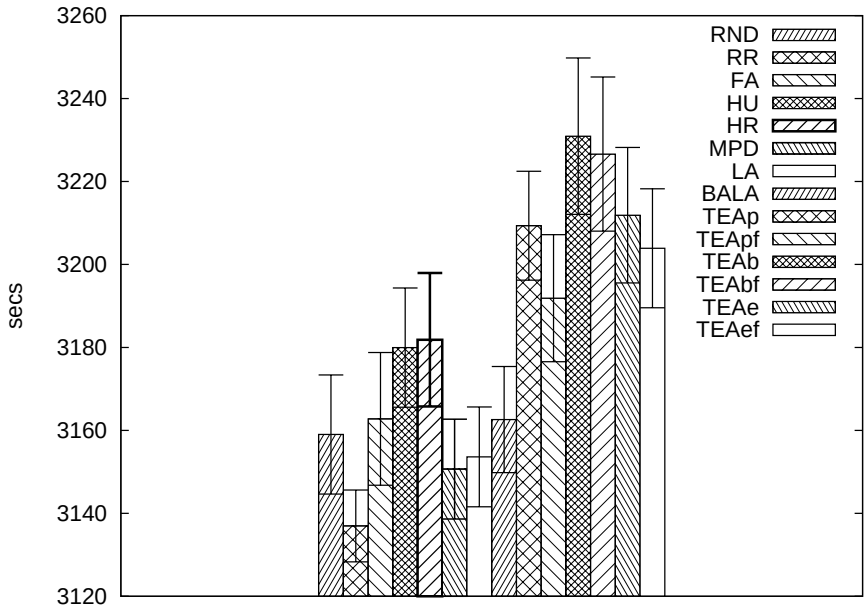

Figura 6.48: Tempo de Processamento de Cargas: DC N.Geo. Het., 10 srvs, K-Ary Fat Tree, Mig, S/SLA, Comp Usr Ind, Mult Pr

A Figura 6.48 traz informações sobre o tempo de processamento das cargas de trabalho após a instanciação da máquina virtual. Neste cenário, avaliamos uma nuvem possuindo 10 servidores distribuídos em um *data center* não geodistribuído e heterogêneo, empregando topologia *K-Ary Fat Tree*. O recurso de migração de máquinas virtuais está ativado, sem SLA de garantia de 100% de processamento para as máquinas virtuais, com comportamento de usuários indefinido e cargas de trabalho com múltiplas prioridades. O algoritmo de escalonamento que apresentou melhores resultados (3.136,96 $\pm$ 8,66 s)

para este quesito foi o RR, empatado tecnicamente pelo menos com o segundo melhor, MPD (3.150,66 ± 12,03 s).

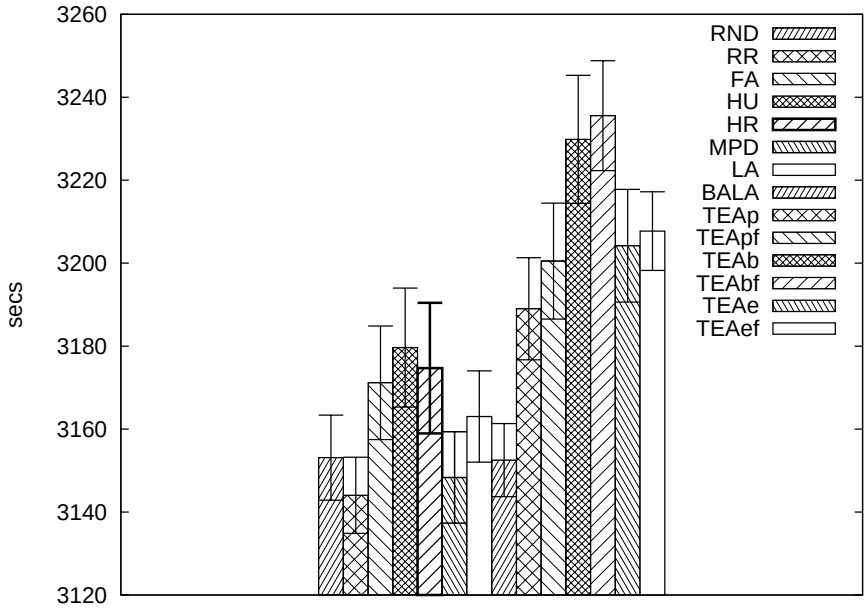

Figura 6.49: Tempo de Processamento de Cargas: DC N.Geo. Het., 10 srvs, Single-Hop Star, Mig, S/SLA, Comp Usr Ind, Mult Pr

Mostramos na Figura 6.49 um gráfico do tempo de processamento das cargas de trabalho após a instanciação da máquina virtual. Neste cenário, avaliamos uma nuvem possuindo 10 servidores distribuídos em um *data center* não geodistribuído e heterogêneo, empregando topologia *Single-Hop Star*. O recurso de migração de máquinas virtuais está ativado, sem SLA de garantia de 100% de processamento para as máquinas virtuais, com comportamento de usuários indefinido e cargas de trabalho com múltiplas prioridades. O algoritmo de escalonamento que apresentou melhores resultados (3.144,04 ± 9,18 s) para este quesito foi o RR, empatado tecnicamente pelo menos com o segundo melhor, MPD (3.148,33 ± 11,02 s).

6.4.7 Análise de Resultados: Tempo de Processamento de Cargas — Prioridade Alta

Temos na Figura 6.50 um histograma sobre o tempo de processamento das cargas de trabalho de alta prioridade após a instanciação da máquina virtual. Neste cenário, avaliamos uma nuvem possuindo 10 servidores distribuídos em dois *data centers* geodistribuídos e homogêneos, empregando topologia *Binary Tree*. O recurso de migração de máquinas virtuais está ativado, sem SLA de garantia de 100% de processamento para as máquinas virtuais, com comportamento de usuários definido e cargas de trabalho com múltiplas prioridades. O algoritmo de escalonamento que apresentou melhores resultados (2.811,80 ± 20,41 s) para este quesito foi o BALA, empatado tecnicamente pelo menos com o segundo melhor, LA (2.812,99 ± 19,99 s).

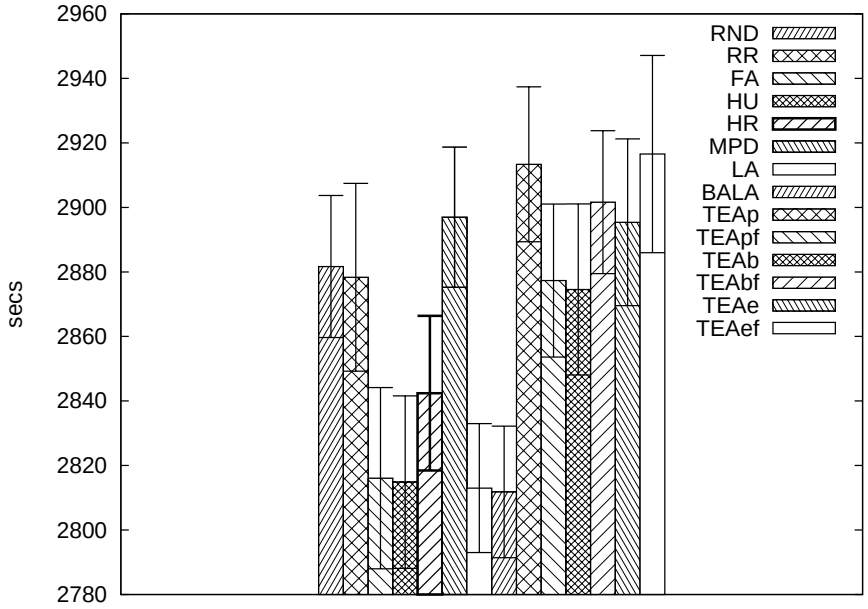

Figura 6.50: Tempo de Processamento de Cargas: Alta Prioridade: DC Geo. Homo., 10 srvs, Binary Tree, Mig, S/SLA, Comp Usr Def, Mult Pr

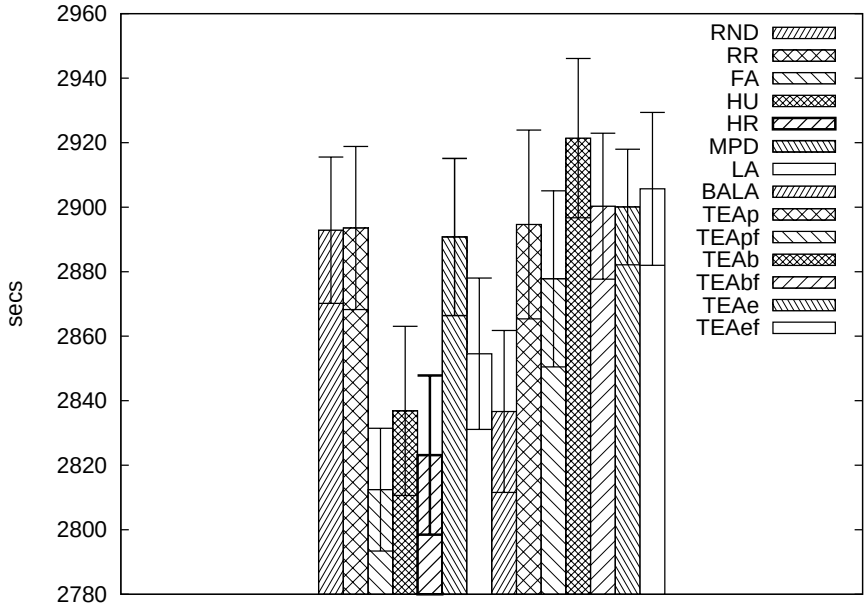

Figura 6.51: Tempo de Processamento de Cargas: Alta Prioridade: DC Geo. Homo., 10 srvs, Flat-Tree, Mig, S/SLA, Comp Usr Def, Mult Pr

A Figura 6.51 traz informações sobre o tempo de processamento das cargas de trabalho de alta prioridade após a instanciação da máquina virtual. Neste cenário, avaliamos uma nuvem possuindo 10 servidores distribuídos em dois *data centers* geodistribuídos e homogêneos, empregando topologia *Flat-Tree*. O recurso de migração de máquinas virtuais está ativado, sem SLA de garantia de 100% de processamento para as máquinas virtuais, com comportamento de usuários definido e cargas de trabalho com múltiplas prioridades. O algoritmo de escalonamento que apresentou melhores resultados (2.812,43

± 19,02 s) para este quesito foi o `FA`, empatado tecnicamente pelo menos com o segundo melhor, `HR` (2.823,15 ± 24,66 s).

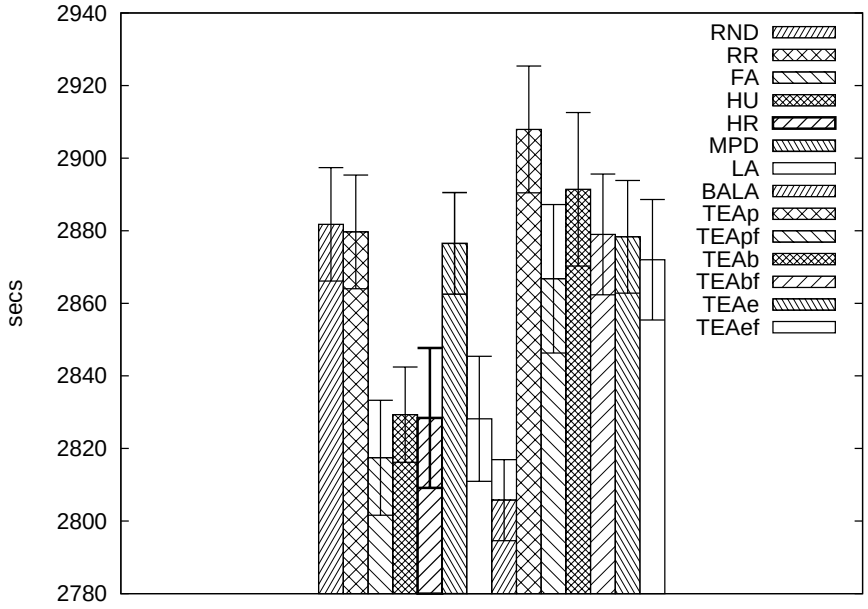

Figura 6.52: Tempo de Processamento de Cargas: Alta Prioridade: DC Geo. Homo., 10 srvs, K-Ary Fat Tree, Mig, S/SLA, Comp Usr Ind, Mult Pr

A Figura 6.52 traz informações sobre o tempo de processamento das cargas de trabalho de alta prioridade após a instanciação da máquina virtual. Neste cenário, avaliamos uma nuvem possuindo 10 servidores distribuídos em dois *data centers* geodistribuídos e homogêneos, empregando topologia *K-Ary Fat Tree*. O recurso de migração de máquinas virtuais está ativado, sem SLA de garantia de 100% de processamento para as máquinas virtuais, com comportamento de usuários indefinido e cargas de trabalho com múltiplas prioridades. O algoritmo de escalonamento que apresentou melhores resultados (2.805,76 ± 11,17 s) para este quesito foi o `BALA`, empatado tecnicamente pelo menos com o segundo melhor, `FA` (2.817,43 ± 15,83 s).

Apresentamos na Figura 6.53 uma comparação do tempo de processamento das cargas de trabalho de alta prioridade após a instanciação da máquina virtual. Neste cenário, avaliamos uma nuvem possuindo 1.000 servidores distribuídos em dois *data centers* geodistribuídos e heterogêneos, empregando topologia *Single-Hop Star*. O recurso de migração de máquinas virtuais está ativado, sem SLA de garantia de 100% de processamento para as máquinas virtuais, com comportamento de usuários indefinido e cargas de trabalho com múltiplas prioridades. O algoritmo de escalonamento que apresentou melhores resultados (3.045,36 ± 0,33 s) para este quesito foi o `LA`, empatado tecnicamente pelo menos com o segundo melhor, `BALA` (3.045,55 ± 0,36 s).

Mostramos na Figura 6.54 um gráfico do tempo de processamento das cargas de trabalho de alta prioridade após a instanciação da máquina virtual. Neste cenário, avaliamos uma nuvem possuindo 100 servidores distribuídos em dois *data centers* geodistribuídos e heterogêneos, empregando topologia *Single-Hop Star*. O recurso de migração de máquinas virtuais está desativado, sem SLA de garantia de 100% de processamento para as

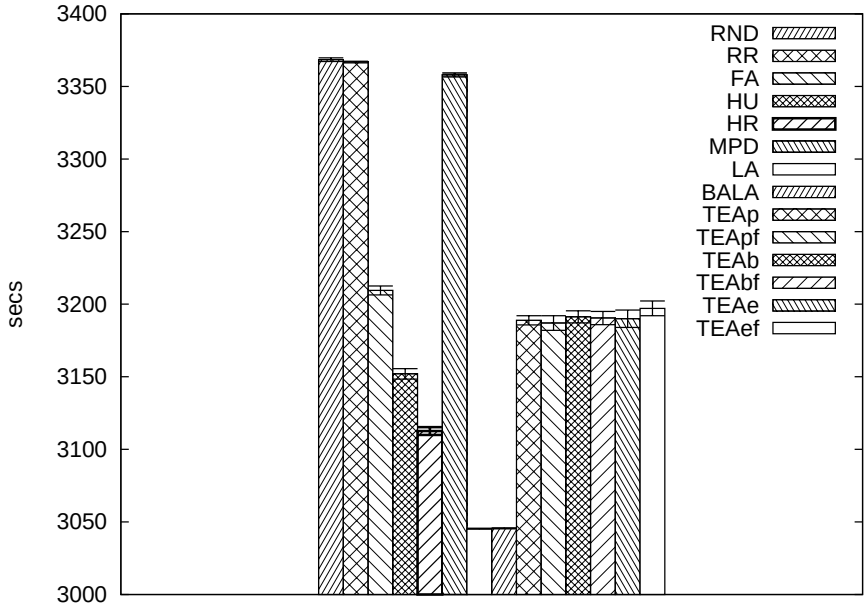

Figura 6.53: Tempo de Processamento de Cargas: Alta Prioridade: DC Geo. Het., 1.000 srvs, Single-Hop Star, Mig, S/SLA, Comp Usr Ind, Mult Pr

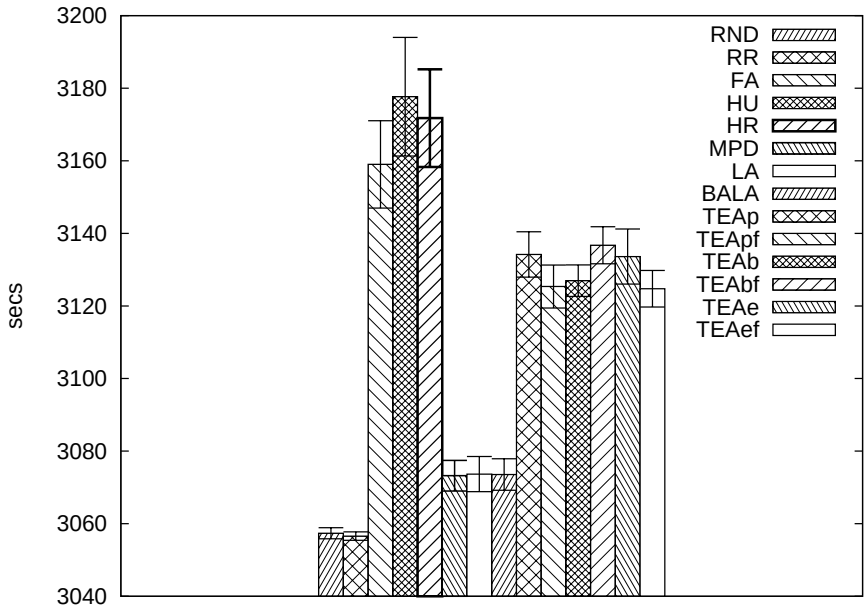

Figura 6.54: Tempo de Processamento de Cargas: Alta Prioridade: DC Geo. Het., 100 srvs, Single-Hop Star, S/Mig, S/SLA, Comp Usr Def, Mult Pr

máquinas virtuais, com comportamento de usuários definido e cargas de trabalho com múltiplas prioridades. O algoritmo de escalonamento que apresentou melhores resultados (3.056,58 ± 1,17 s) para este quesito foi o `RR`, empatado tecnicamente pelo menos com o segundo melhor, `RND` (3.057,35 ± 1,53 s).

Apresentamos na Figura 6.55 uma comparação do tempo de processamento das cargas de trabalho de alta prioridade após a instanciação da máquina virtual. Neste cenário, avaliamos uma nuvem possuindo 10 servidores distribuídos em um *data center* não geo-

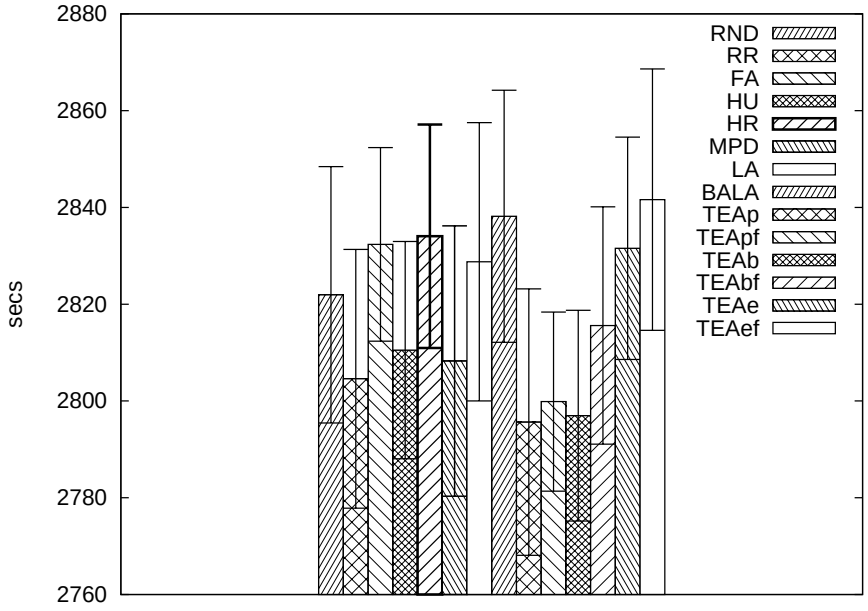

Figura 6.55: Tempo de Processamento de Cargas: Alta Prioridade: DC N.Geo. Homo., 10 srvs, Binary Tree, S/Mig, S/SLA, Comp Usr Def, Mult Pr

distribuído e homogêneo, empregando topologia *Binary Tree*. O recurso de migração de máquinas virtuais está desativado, sem SLA de garantia de 100% de processamento para as máquinas virtuais, com comportamento de usuários definido e cargas de trabalho com múltiplas prioridades. O algoritmo de escalonamento que apresentou melhores resultados (2.795,63 ± 27,53 s) para este quesito foi o `TEA` (`TEAp`), empatado tecnicamente pelo menos com o segundo melhor, `RR` (2.804,59 ± 26,74 s).

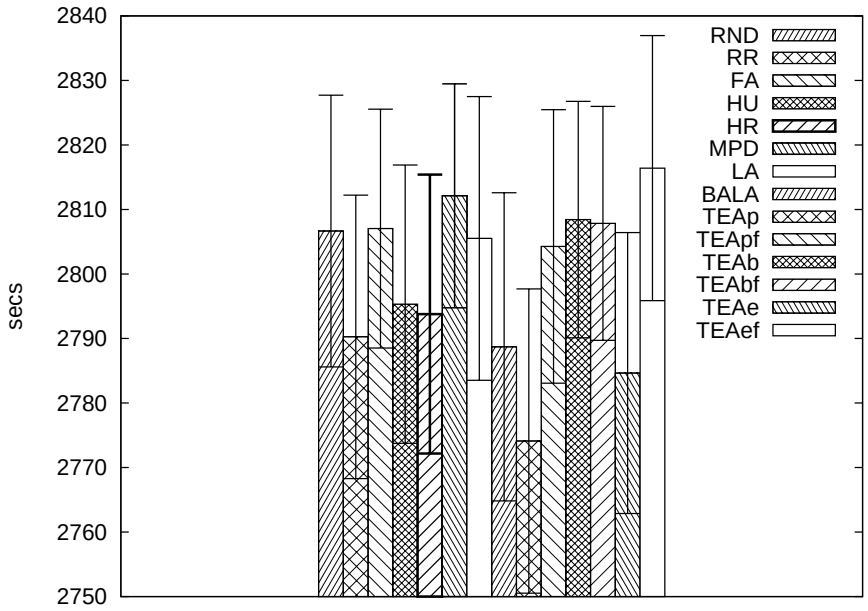

Figura 6.56: Tempo de Processamento de Cargas: Alta Prioridade: DC N.Geo. Het., 10 srvs, K-Ary Fat Tree, S/Mig, SLA, Comp Usr Def, Mult Pr

Apresentamos na Figura 6.56 uma comparação do tempo de processamento das cargas de trabalho de alta prioridade após a instanciação da máquina virtual. Neste cenário, avaliamos uma nuvem possuindo 10 servidores distribuídos em um *data center* não geo-distribuído e heterogêneo, empregando topologia *K-Ary Fat Tree*. O recurso de migração de máquinas virtuais está desativado, com SLA de garantia de 100% de processamento para as máquinas virtuais, com comportamento de usuários definido e cargas de trabalho com múltiplas prioridades. O algoritmo de escalonamento que apresentou melhores resultados (2.774,11 ± 23,59 s) para este quesito foi o `TEA` (`TEAp`), empatado tecnicamente pelo menos com o segundo melhor, `BALA` (2.788,70 ± 23,88 s).

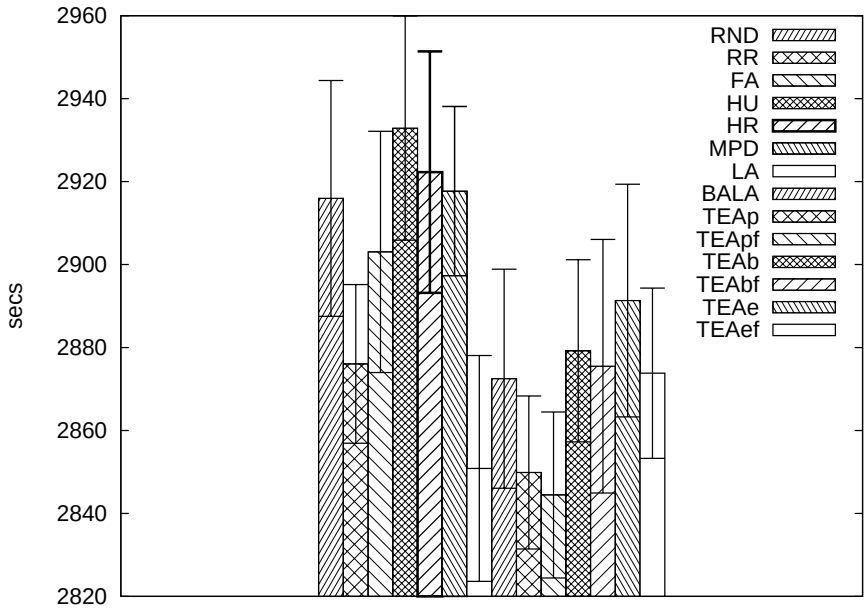

Figura 6.57: Tempo de Processamento de Cargas: Alta Prioridade: DC N.Geo. Het., 10 srvs, Single-Hop Star, Mig, S/SLA, Comp Usr Def, Mult Pr

Temos na Figura 6.57 um histograma sobre o tempo de processamento das cargas de trabalho de alta prioridade após a instanciação da máquina virtual. Neste cenário, avaliamos uma nuvem possuindo 10 servidores distribuídos em um *data center* não geo-distribuído e heterogêneo, empregando topologia *Single-Hop Star*. O recurso de migração de máquinas virtuais está ativado, sem SLA de garantia de 100% de processamento para as máquinas virtuais, com comportamento de usuários definido e cargas de trabalho com múltiplas prioridades. O algoritmo de escalonamento que apresentou melhores resultados (2.844,45 ± 20,02 s) para este quesito foi o `TEA` (`TEApf`), empatado tecnicamente pelo menos com o segundo melhor, `LA` (2.850,85 ± 27,23 s).

Temos na Figura 6.58 um histograma sobre o tempo de processamento das cargas de trabalho de alta prioridade após a instanciação da máquina virtual. Neste cenário, avaliamos uma nuvem possuindo 10 servidores distribuídos em um *data center* não geo-distribuído e heterogêneo, empregando topologia *Single-Hop Star*. O recurso de migração de máquinas virtuais está ativado, com SLA de garantia de 100% de processamento para as máquinas virtuais, com comportamento de usuários definido e cargas de trabalho com múltiplas prioridades. O algoritmo de escalonamento que apresentou melhores resultados

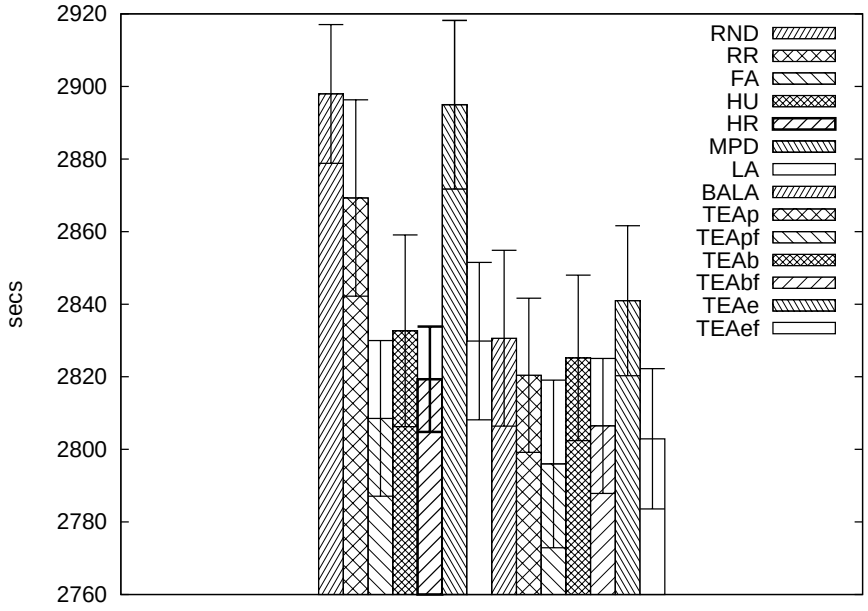

Figura 6.58: Tempo de Processamento de Cargas: Alta Prioridade: DC N.Geo. Het., 10 srvs, Single-Hop Star, Mig, SLA, Comp Usr Def, Mult Pr

(2.795,98 $\pm$ 23,11 s) para este quesito foi o `TEA` (`TEApf`), empatado tecnicamente pelo menos com o segundo melhor, `FA` (2.808,52 $\pm$ 21,44 s).

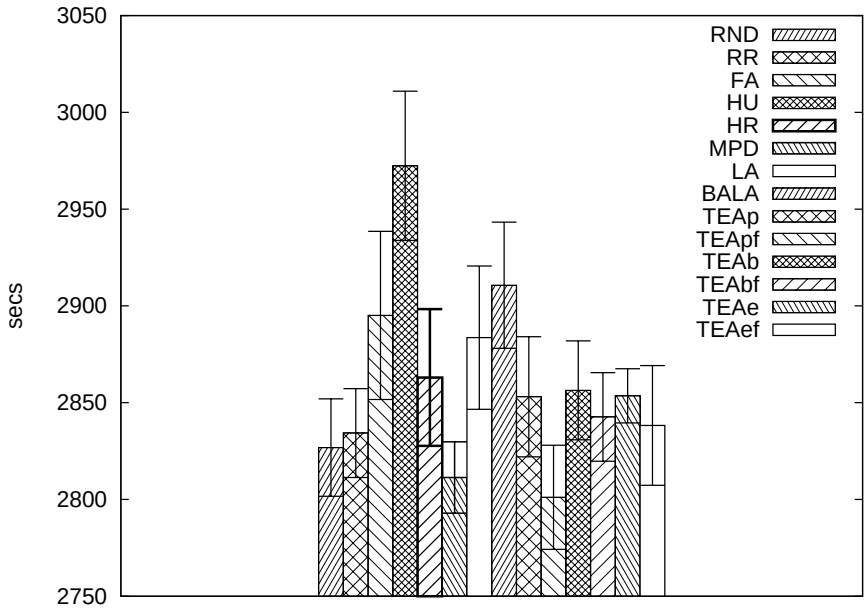

Figura 6.59: Tempo de Processamento de Cargas: Alta Prioridade: DC N.Geo. Het., 10 srvs, Single-Hop Star, S/Mig, S/SLA, Comp Usr Def, Mult Pr

A Figura 6.59 traz informações sobre o tempo de processamento das cargas de trabalho de alta prioridade após a instanciação da máquina virtual. Neste cenário, avaliamos uma nuvem possuindo 10 servidores distribuídos em um *data center* não geodistribuído e heterogêneo, empregando topologia *Single-Hop Star*. O recurso de migração de máquinas

virtuais está desativado, sem SLA de garantia de 100% de processamento para as máquinas virtuais, com comportamento de usuários definido e cargas de trabalho com múltiplas prioridades. O algoritmo de escalonamento que apresentou melhores resultados (2.801,10 ± 26,90 s) para este quesito foi o `TEA` (`TEApf`), empatado tecnicamente pelo menos com o segundo melhor, `MPD` (2.811,35 ± 18,41 s).

6.4.8 Análise de Resultados: Tempo de Processamento de Cargas — Prioridade Normal

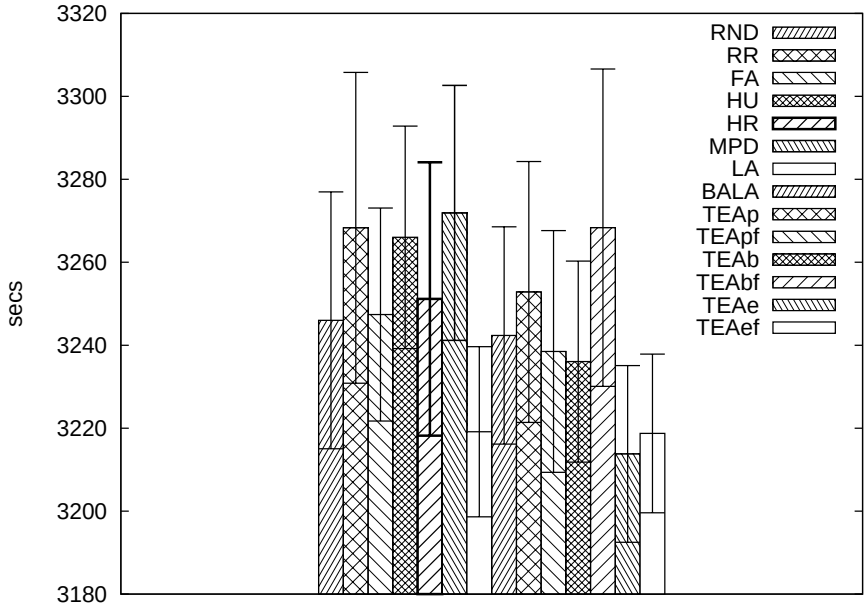

Figura 6.60: Tempo de Processamento de Cargas: Prioridade Normal: DC Geo. Het., 10 srvs, K-Ary Fat Tree, S/Mig, SLA, Comp Usr Def, Mult Pr

Temos na Figura 6.60 um histograma sobre o tempo de processamento das cargas de trabalho de prioridade normal após a instanciação da máquina virtual. Neste cenário, avaliamos uma nuvem possuindo 10 servidores distribuídos em dois *data centers* geodistribuídos e heterogêneos, empregando topologia *K-Ary Fat Tree*. O recurso de migração de máquinas virtuais está desativado, com SLA de garantia de 100% de processamento para as máquinas virtuais, com comportamento de usuários definido e cargas de trabalho com múltiplas prioridades. O algoritmo de escalonamento que apresentou melhores resultados (3.213,80 ± 21,30 s) para este quesito foi o `TEA` (`TEAe`), empatado tecnicamente pelo menos com o segundo melhor, `LA` (3.219,13 ± 20,52 s).

A Figura 6.61 traz informações sobre o tempo de processamento das cargas de trabalho de prioridade normal após a instanciação da máquina virtual. Neste cenário, avaliamos uma nuvem possuindo 1.000 servidores distribuídos em dois *data centers* geodistribuídos e homogêneos, empregando topologia *Single-Hop Star*. O recurso de migração de máquinas virtuais está ativado, sem SLA de garantia de 100% de processamento para as máquinas virtuais, com comportamento de usuários indefinido e cargas de trabalho com múltiplas prioridades. O algoritmo de escalonamento que apresentou melhores resultados (3.047,47

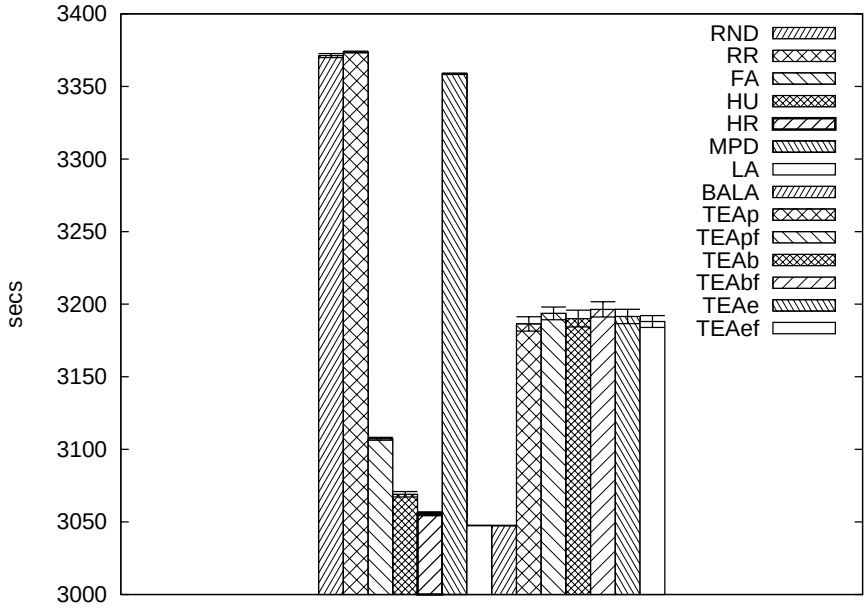

Figura 6.61: Tempo de Processamento de Cargas: Prioridade Normal: DC Geo. Homo., 1.000 srvs, Single-Hop Star, Mig, S/SLA, Comp Usr Ind, Mult Pr

$\pm$ 0,17 s) para este quesito foi o `BALA`, empatado tecnicamente pelo menos com o segundo melhor, `LA` (3.047,57 $\pm$ 0,13 s).

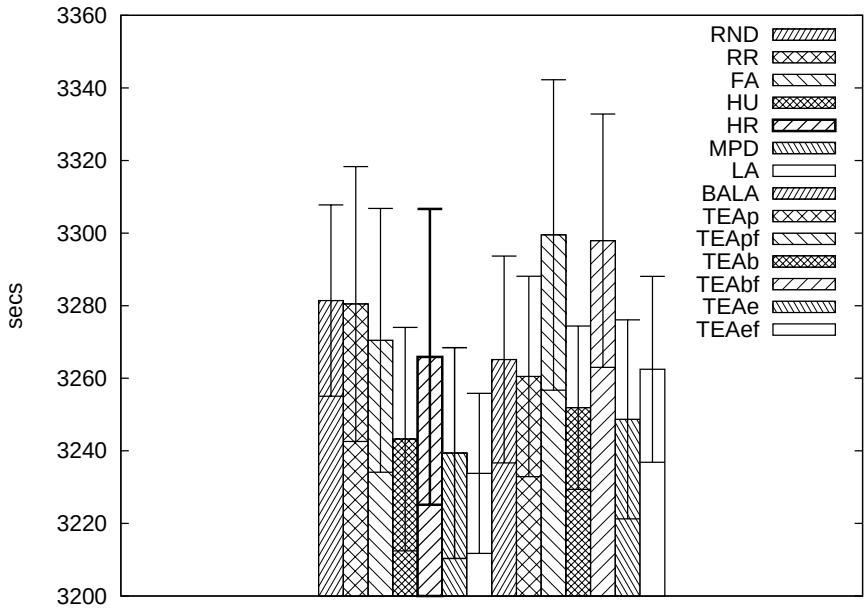

Figura 6.62: Tempo de Processamento de Cargas: Prioridade Normal: DC Geo. Homo., 10 srvs, Single-Hop Star, Mig, SLA, Comp Usr Def, Mult Pr

Mostramos na Figura 6.62 um gráfico do tempo de processamento das cargas de trabalho de prioridade normal após a instanciação da máquina virtual. Neste cenário, avaliamos uma nuvem possuindo 10 servidores distribuídos em dois *data centers* geodistribuídos e homogêneos, empregando topologia *Single-Hop Star*. O recurso de migração de máquinas

virtuais está ativado, com SLA de garantia de 100% de processamento para as máquinas virtuais, com comportamento de usuários definido e cargas de trabalho com múltiplas prioridades. O algoritmo de escalonamento que apresentou melhores resultados (3.233,78 ± 22,06 s) para este quesito foi o `LA`, empatado tecnicamente pelo menos com o segundo melhor, `MPD` (3.239,38 ± 29,04 s).

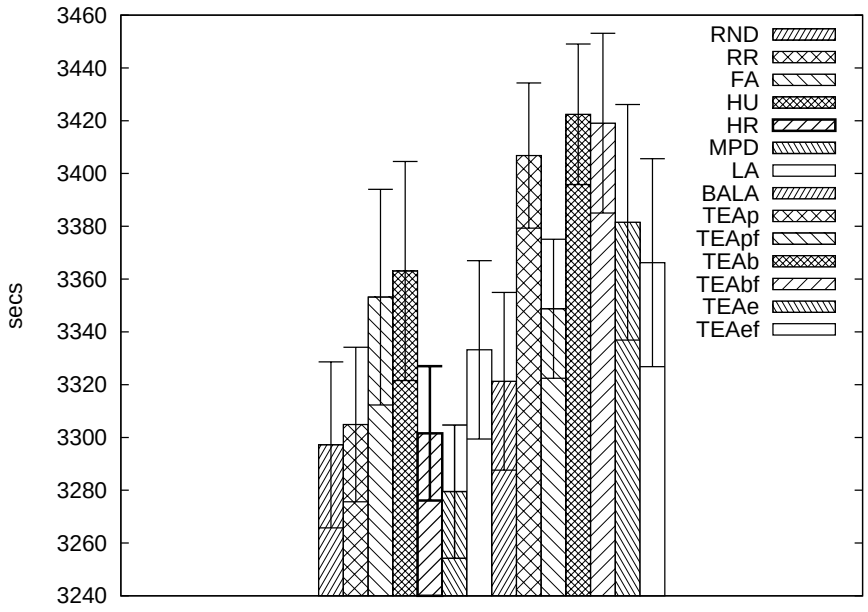

Figura 6.63: Tempo de Processamento de Cargas: Prioridade Normal: DC N.Geo. Het., 10 srvs, Binary Tree, Mig, S/SLA, Comp Usr Def, Mult Pr

Temos na Figura 6.63 um histograma sobre o tempo de processamento das cargas de trabalho de prioridade normal após a instanciação da máquina virtual. Neste cenário, avaliamos uma nuvem possuindo 10 servidores distribuídos em um *data center* não geodistribuído e heterogêneo, empregando topologia *Binary Tree*. O recurso de migração de máquinas virtuais está ativado, sem SLA de garantia de 100% de processamento para as máquinas virtuais, com comportamento de usuários definido e cargas de trabalho com múltiplas prioridades. O algoritmo de escalonamento que apresentou melhores resultados (3.279,48 ± 25,24 s) para este quesito foi o `MPD`, empatado tecnicamente pelo menos com o segundo melhor, `RND` (3.297,20 ± 31,44 s).

Mostramos na Figura 6.64 um gráfico do tempo de processamento das cargas de trabalho de prioridade normal após a instanciação da máquina virtual. Neste cenário, avaliamos uma nuvem possuindo 10 servidores distribuídos em um *data center* não geodistribuído e heterogêneo, empregando topologia *Binary Tree*. O recurso de migração de máquinas virtuais está ativado, com SLA de garantia de 100% de processamento para as máquinas virtuais, com comportamento de usuários definido e cargas de trabalho com múltiplas prioridades. O algoritmo de escalonamento que apresentou melhores resultados (3.239,74 ± 24,72 s) para este quesito foi o `TEA` (`TEAbf`), empatado tecnicamente pelo menos com o segundo melhor, `HR` (3.243,90 ± 28,26 s).

Temos na Figura 6.65 um histograma sobre o tempo de processamento das cargas de trabalho de prioridade normal após a instanciação da máquina virtual. Neste cenário,

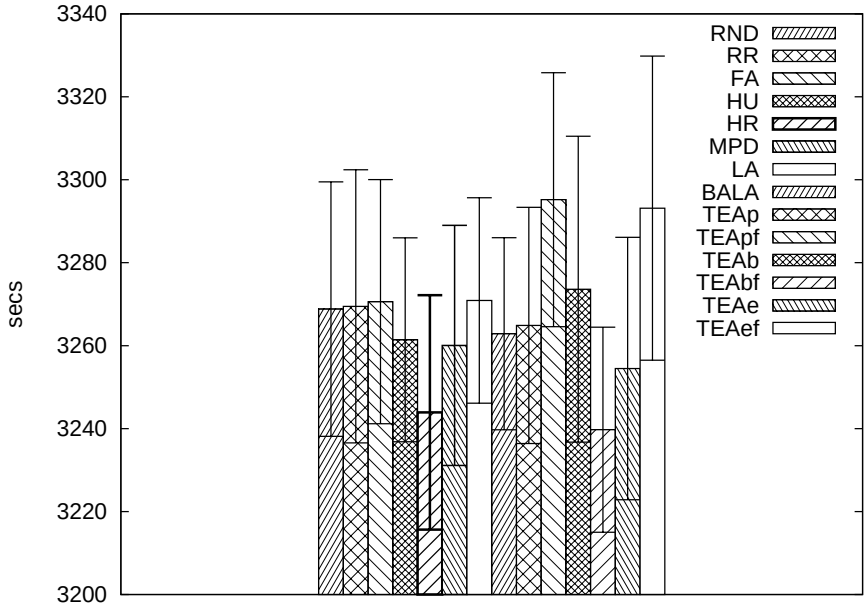

Figura 6.64: Tempo de Processamento de Cargas: Prioridade Normal: DC N.Geo. Het., 10 srvs, Binary Tree, Mig, SLA, Comp Usr Def, Mult Pr

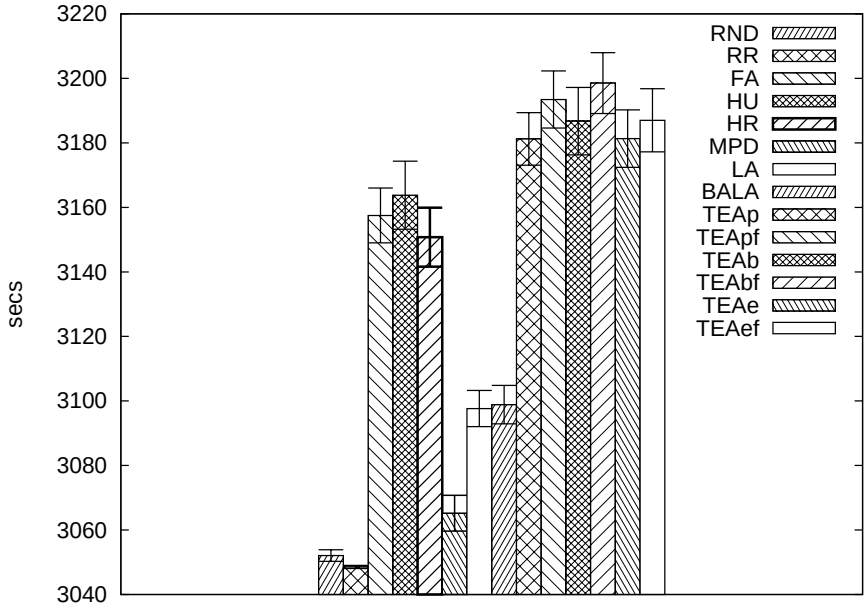

Figura 6.65: Tempo de Processamento de Cargas: Prioridade Normal: DC N.Geo. Het., 100 srvs, Flat-Tree, Mig, S/SLA, Comp Usr Ind, Mult Pr

avaliamos uma nuvem possuindo 100 servidores distribuídos em um *data center* não geodistribuído e heterogêneo, empregando topologia *Flat-Tree*. O recurso de migração de máquinas virtuais está ativado, sem SLA de garantia de 100% de processamento para as máquinas virtuais, com comportamento de usuários indefinido e cargas de trabalho com múltiplas prioridades. O algoritmo de escalonamento que apresentou melhores resultados (3.048,54 $\pm$ 0,50 s) para este quesito foi o `RR`, apresentando, considerando o intervalo de confiança, isoladamente os melhores valores do parâmetro comparado dentre todos os

algoritmos para este cenário, com uma melhora variando de 0,0% a 0,2% comparado ao segundo melhor (RND).

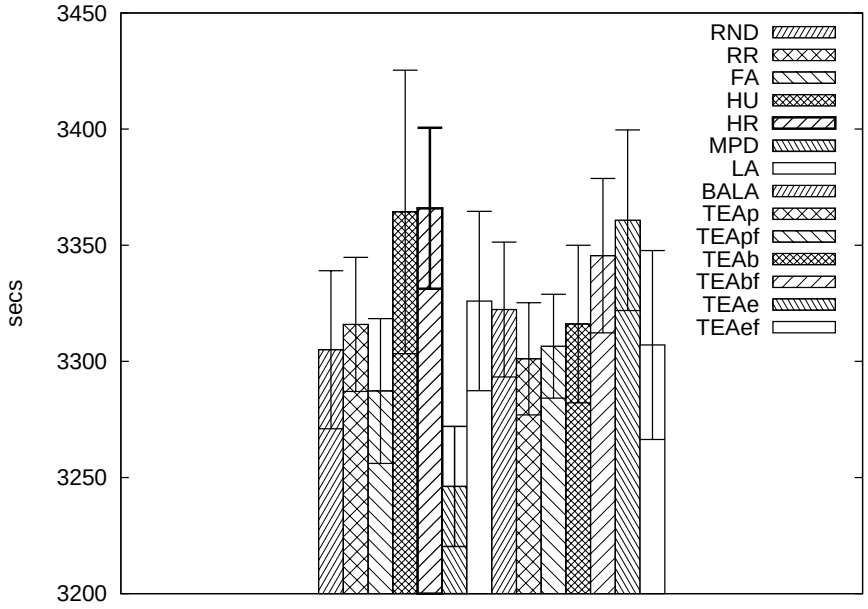

Figura 6.66: Tempo de Processamento de Cargas: Prioridade Normal: DC N.Geo. Het., 10 srvs, Flat-Tree, S/Mig, S/SLA, Comp Usr Def, Mult Pr

Mostramos na Figura 6.66 um gráfico do tempo de processamento das cargas de trabalho de prioridade normal após a instanciação da máquina virtual. Neste cenário, avaliamos uma nuvem possuindo 10 servidores distribuídos em um *data center* não geodistribuído e heterogêneo, empregando topologia *Flat-Tree*. O recurso de migração de máquinas virtuais está desativado, sem SLA de garantia de 100% de processamento para as máquinas virtuais, com comportamento de usuários definido e cargas de trabalho com múltiplas prioridades. O algoritmo de escalonamento que apresentou melhores resultados (3.246,17 ± 25,85 s) para este quesito foi o MPD, empatado tecnicamente pelo menos com o segundo melhor, FA (3.287,25 ± 31,19 s).

Apresentamos na Figura 6.67 uma comparação do tempo de processamento das cargas de trabalho de prioridade normal após a instanciação da máquina virtual. Neste cenário, avaliamos uma nuvem possuindo 10 servidores distribuídos em um *data center* não geodistribuído e heterogêneo, empregando topologia *Flat-Tree*. O recurso de migração de máquinas virtuais está desativado, sem SLA de garantia de 100% de processamento para as máquinas virtuais, com comportamento de usuários indefinido e cargas de trabalho com somente uma prioridade. O algoritmo de escalonamento que apresentou melhores resultados (3.083,46 ± 2,75 s) para este quesito foi o RR, empatado tecnicamente pelo menos com o segundo melhor, MPD (3.094,99 ± 11,12 s).

Temos na Figura 6.68 um histograma sobre o tempo de processamento das cargas de trabalho de prioridade normal após a instanciação da máquina virtual. Neste cenário, avaliamos uma nuvem possuindo 10 servidores distribuídos em um *data center* não geodistribuído e heterogêneo, empregando topologia *Single-Hop Star*. O recurso de migração de máquinas virtuais está ativado, sem SLA de garantia de 100% de processamento para

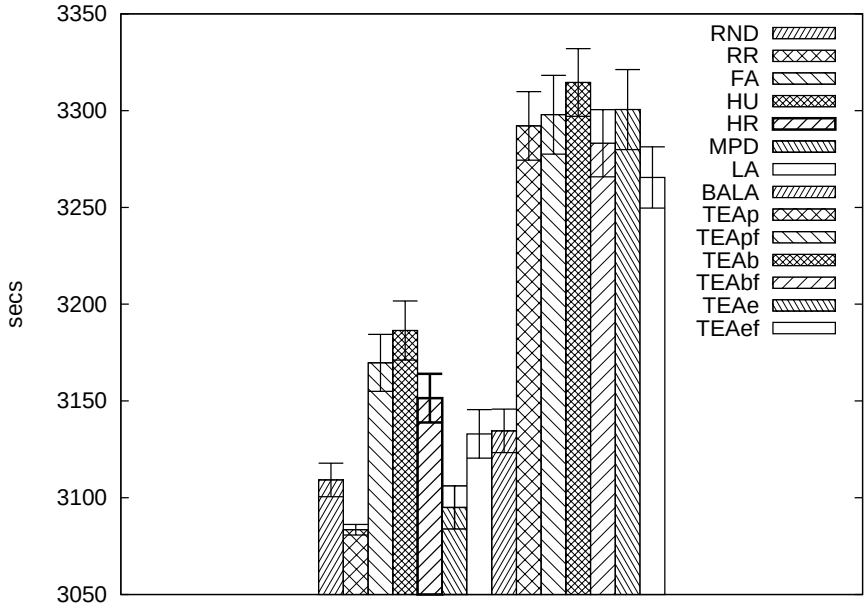

Figura 6.67: Tempo de Processamento de Cargas: Prioridade Normal: DC N.Geo. Het., 10 srvs, Flat-Tree, S/Mig, S/SLA, Comp Usr Ind, S/Mult Pr

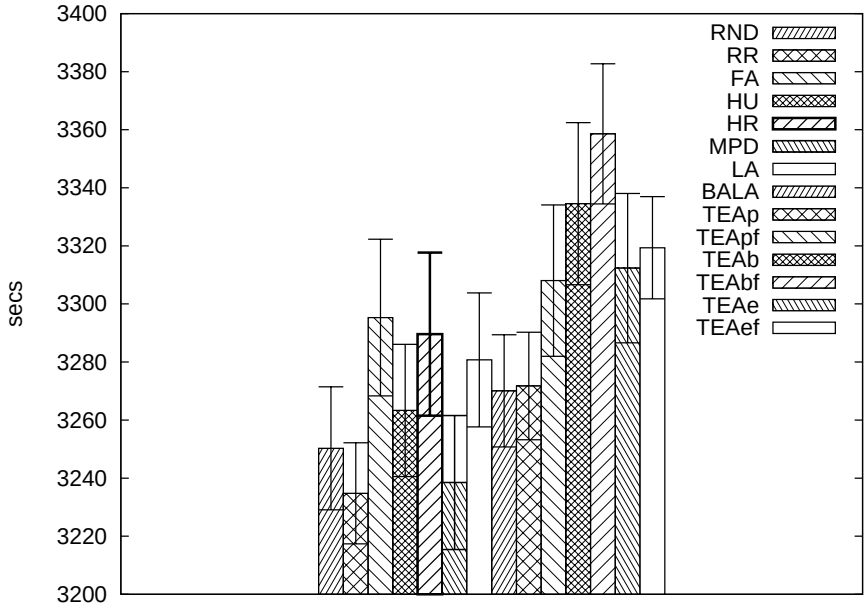

Figura 6.68: Tempo de Processamento de Cargas: Prioridade Normal: DC N.Geo. Het., 10 srvs, Single-Hop Star, Mig, S/SLA, Comp Usr Ind, Mult Pr

as máquinas virtuais, com comportamento de usuários indefinido e cargas de trabalho com múltiplas prioridades. O algoritmo de escalonamento que apresentou melhores resultados (3.234,78 $\pm$ 17,40 s) para este quesito foi o `RR`, empatado tecnicamente pelo menos com o segundo melhor, `MPD` (3.238,49 $\pm$ 23,06 s).

6.4.9 Análise de Resultados: Tempo de Processamento de Cargas — Prioridade Baixa

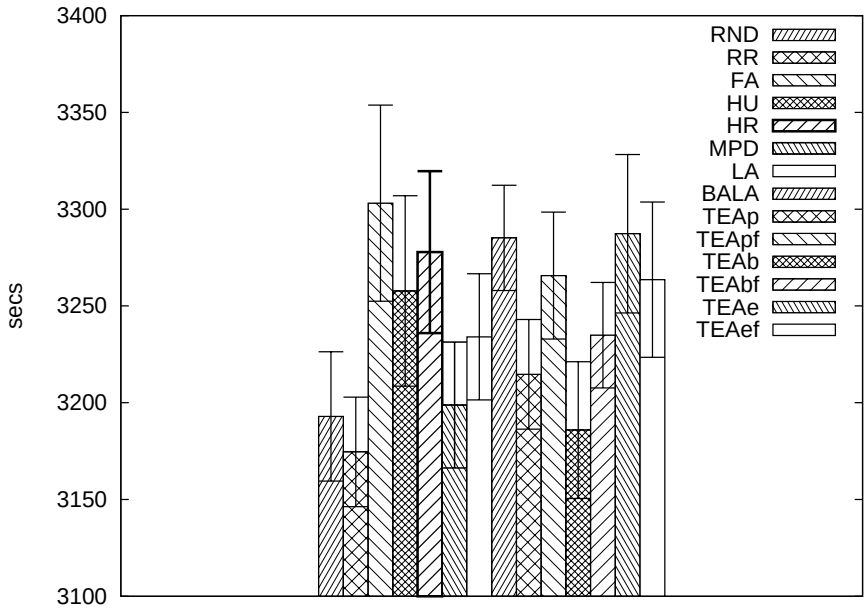

Figura 6.69: Tempo de Processamento de Cargas: Baixa Prioridade: DC Geo. Het., 10 srvs, Binary Tree, S/Mig, S/SLA, Comp Usr Def, Mult Pr

Mostramos na Figura 6.69 um gráfico do tempo de processamento das cargas de trabalho de baixa prioridade após a instanciação da máquina virtual. Neste cenário, avaliamos uma nuvem possuindo 10 servidores distribuídos em dois *data centers* geodistribuídos e heterogêneos, empregando topologia *Binary Tree*. O recurso de migração de máquinas virtuais está desativado, sem SLA de garantia de 100% de processamento para as máquinas virtuais, com comportamento de usuários definido e cargas de trabalho com múltiplas prioridades. O algoritmo de escalonamento que apresentou melhores resultados (3.174,59 ± 28,28 s) para este quesito foi o `RR`, empatado tecnicamente pelo menos com o segundo melhor, `TEAb` (3.185,86 ± 35,30 s).

Mostramos na Figura 6.70 um gráfico do tempo de processamento das cargas de trabalho de baixa prioridade após a instanciação da máquina virtual. Neste cenário, avaliamos uma nuvem possuindo 10 servidores distribuídos em dois *data centers* geodistribuídos e heterogêneos, empregando topologia *Flat-Tree*. O recurso de migração de máquinas virtuais está desativado, sem SLA de garantia de 100% de processamento para as máquinas virtuais, com comportamento de usuários definido e cargas de trabalho com múltiplas prioridades. O algoritmo de escalonamento que apresentou melhores resultados (3.159,41 ± 45,15 s) para este quesito foi o `MPD`, empatado tecnicamente pelo menos com o segundo melhor, `RND` (3.190,99 ± 41,87 s).

Apresentamos na Figura 6.71 uma comparação do tempo de processamento das cargas de trabalho de baixa prioridade após a instanciação da máquina virtual. Neste cenário, avaliamos uma nuvem possuindo 10 servidores distribuídos em dois *data centers* geodistribuídos e heterogêneos, empregando topologia *Single-Hop Star*. O recurso de migração de

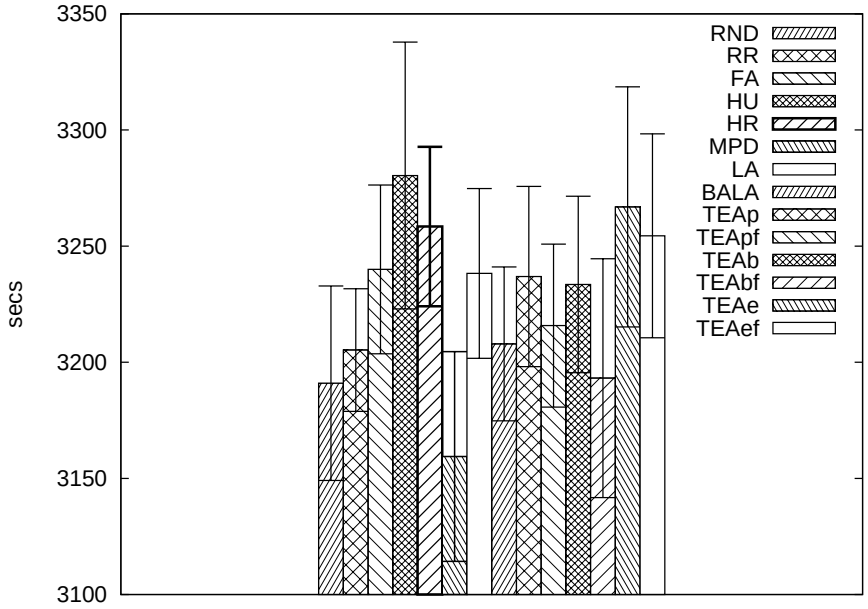

Figura 6.70: Tempo de Processamento de Cargas: Baixa Prioridade: DC Geo. Het., 10 srvs, Flat-Tree, S/Mig, S/SLA, Comp Usr Def, Mult Pr

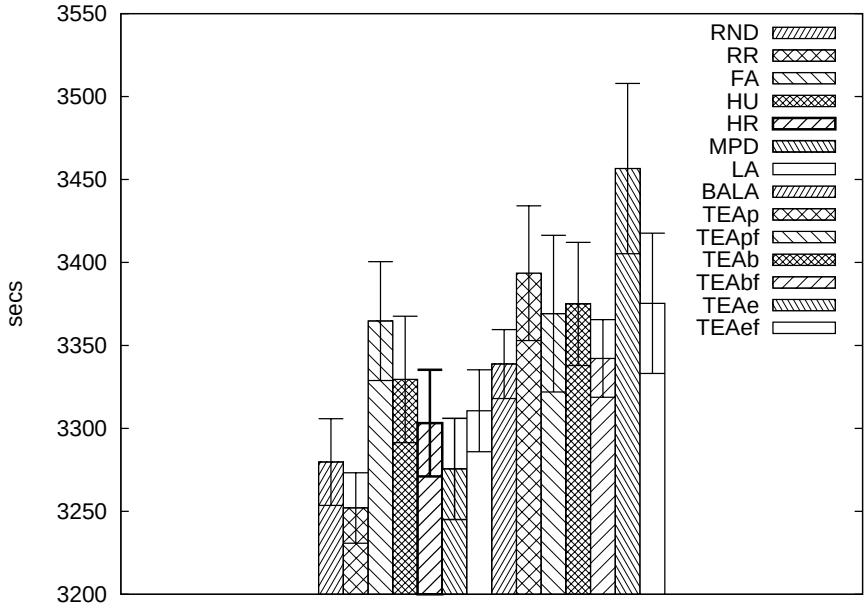

Figura 6.71: Tempo de Processamento de Cargas: Baixa Prioridade: DC Geo. Het., 10 srvs, Single-Hop Star, Mig, S/SLA, Comp Usr Ind, Mult Pr

máquinas virtuais está ativado, sem SLA de garantia de 100% de processamento para as máquinas virtuais, com comportamento de usuários indefinido e cargas de trabalho com múltiplas prioridades. O algoritmo de escalonamento que apresentou melhores resultados (3.252,02 $\pm$ 21,23 s) para este quesito foi o `RR`, empatado tecnicamente pelo menos com o segundo melhor, `MPD` (3.275,57 $\pm$ 30,52 s).

A Figura 6.72 traz informações sobre o tempo de processamento das cargas de trabalho de baixa prioridade após a instanciação da máquina virtual. Neste cenário, avaliamos uma

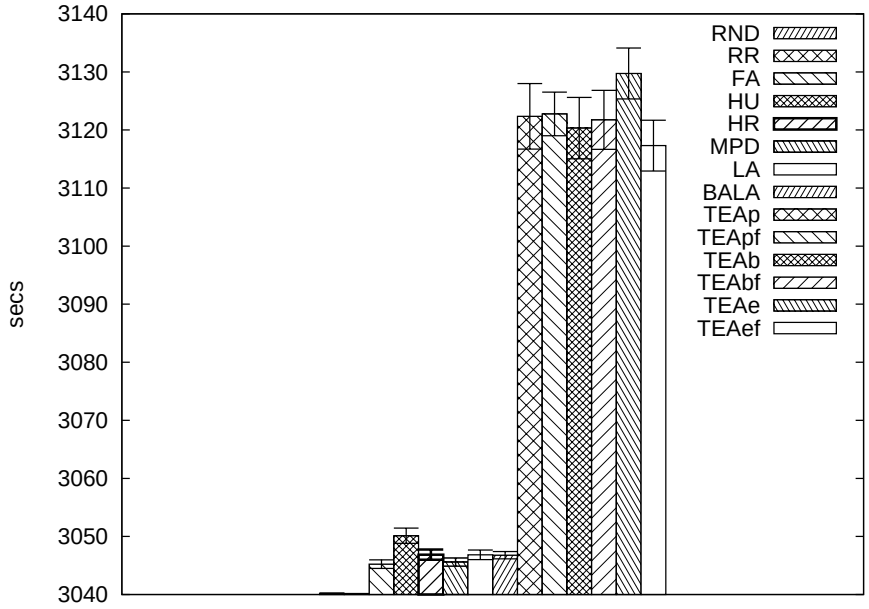

Figura 6.72: Tempo de Processamento de Cargas: Baixa Prioridade: DC Geo. Homo., 1.000 srvs, Single-Hop Star, S/Mig, S/SLA, Comp Usr Def, Mult Pr

nuvem possuindo 1.000 servidores distribuídos em dois *data centers* geodistribuídos e homogêneos, empregando topologia *Single-Hop Star*. O recurso de migração de máquinas virtuais está desativado, sem SLA de garantia de 100% de processamento para as máquinas virtuais, com comportamento de usuários definido e cargas de trabalho com múltiplas prioridades. O algoritmo de escalonamento que apresentou melhores resultados (3.040,15 ± 0,03 s) para este quesito foi o `RR`, empatado tecnicamente pelo menos com o segundo melhor, `RND` (3.040,20 ± 0,07 s).

A Figura 6.73 traz informações sobre o tempo de processamento das cargas de trabalho de baixa prioridade após a instanciação da máquina virtual. Neste cenário, avaliamos uma nuvem possuindo 10 servidores distribuídos em um *data center* não geodistribuído e homogêneo, empregando topologia *Binary Tree*. O recurso de migração de máquinas virtuais está desativado, com SLA de garantia de 100% de processamento para as máquinas virtuais, com comportamento de usuários definido e cargas de trabalho com múltiplas prioridades. O algoritmo de escalonamento que apresentou melhores resultados (3.129,25 ± 28,97 s) para este quesito foi o `TEA` (`TEAb`), empatado tecnicamente pelo menos com o segundo melhor, `RND` (3.155,06 ± 24,70 s).

A Figura 6.74 traz informações sobre o tempo de processamento das cargas de trabalho de baixa prioridade após a instanciação da máquina virtual. Neste cenário, avaliamos uma nuvem possuindo 100 servidores distribuídos em um *data center* não geodistribuído e heterogêneo, empregando topologia *Flat-Tree*. O recurso de migração de máquinas virtuais está ativado, sem SLA de garantia de 100% de processamento para as máquinas virtuais, com comportamento de usuários indefinido e cargas de trabalho com múltiplas prioridades. O algoritmo de escalonamento que apresentou melhores resultados (3.071,78 ± 0,70 s) para este quesito foi o `RR`, apresentando, considerando o intervalo de confiança, isoladamente os melhores valores do parâmetro comparado dentre todos os algoritmos

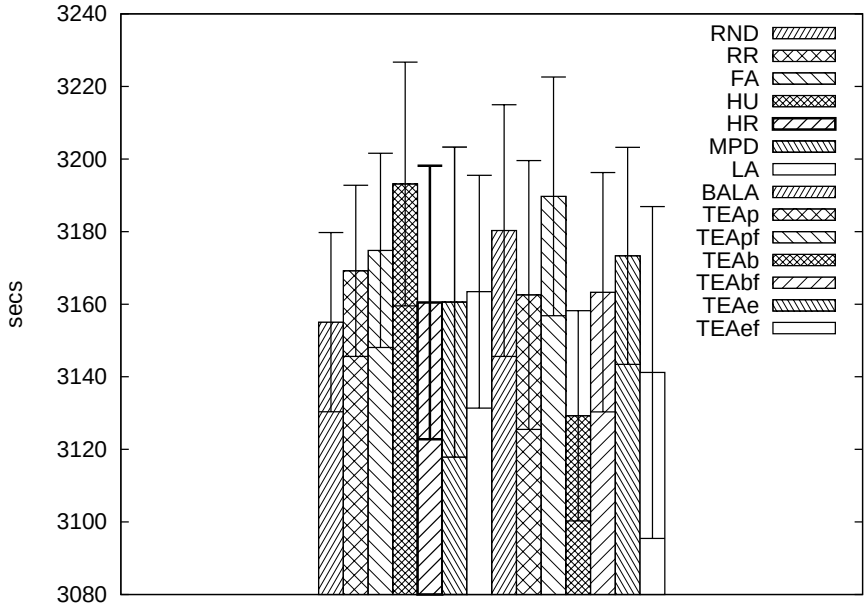

Figura 6.73: Tempo de Processamento de Cargas: Baixa Prioridade: DC N.Geo. Homo., 10 srvs, Binary Tree, S/Mig, SLA, Comp Usr Def, Mult Pr

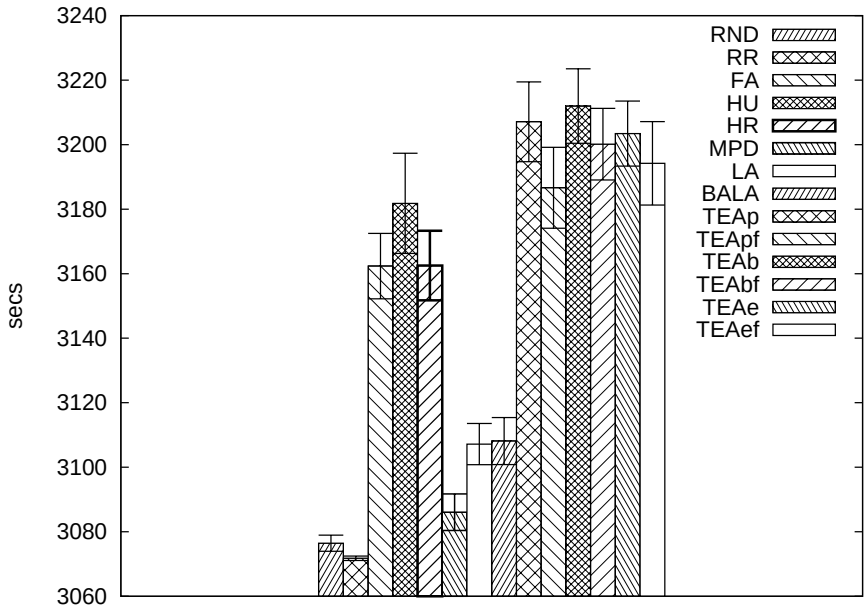

Figura 6.74: Tempo de Processamento de Cargas: Baixa Prioridade: DC N.Geo. Het., 100 srvs, Flat-Tree, Mig, S/SLA, Comp Usr Ind, Mult Pr

para este cenário, com uma melhora variando de 0,0% a 0,3% comparado ao segundo melhor (`RND`).

A Figura 6.75 traz informações sobre o tempo de processamento das cargas de trabalho de baixa prioridade após a instanciação da máquina virtual. Neste cenário, avaliamos uma nuvem possuindo 10 servidores distribuídos em um *data center* não geodistribuído e homogêneo, empregando topologia *Flat-Tree*. O recurso de migração de máquinas virtuais está desativado, sem SLA de garantia de 100% de processamento para as máquinas

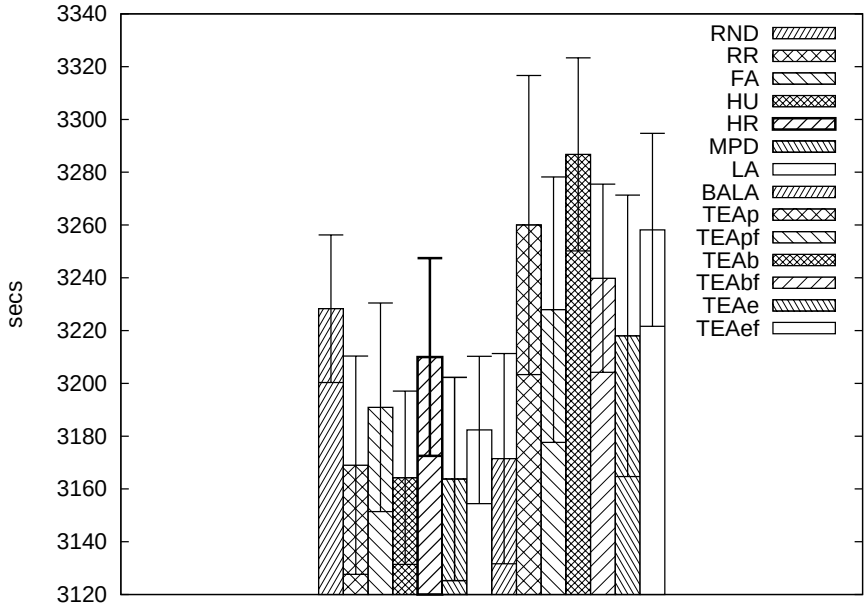

Figura 6.75: Tempo de Processamento de Cargas: Baixa Prioridade: DC N.Geo. Homo., 10 srvs, Flat-Tree, S/Mig, S/SLA, Comp Usr Def, Mult Pr

virtuais, com comportamento de usuários definido e cargas de trabalho com múltiplas prioridades. O algoritmo de escalonamento que apresentou melhores resultados (3.163,77 ± 38,52 s) para este quesito foi o `MPD`, empatado tecnicamente pelo menos com o segundo melhor, `HU` (3.164,23 ± 32,84 s).

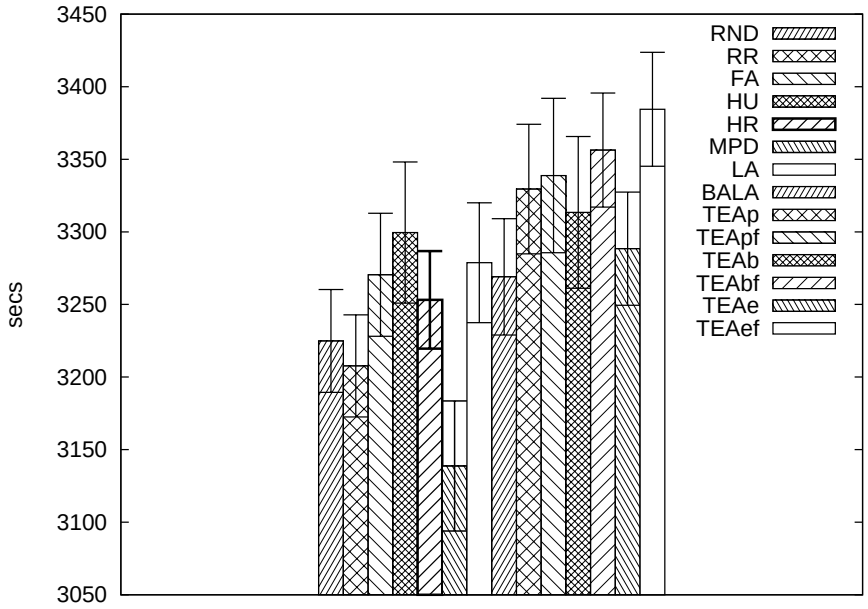

Figura 6.76: Tempo de Processamento de Cargas: Baixa Prioridade: DC N.Geo. Het., 10 srvs, K-Ary Fat Tree, Mig, S/SLA, Comp Usr Def, Mult Pr

Temos na Figura 6.76 um histograma sobre o tempo de processamento das cargas de trabalho de baixa prioridade após a instanciação da máquina virtual. Neste cenário,

avaliamos uma nuvem possuindo 10 servidores distribuídos em um *data center* não geodistribuído e heterogêneo, empregando topologia *K-Ary Fat Tree*. O recurso de migração de máquinas virtuais está ativado, sem SLA de garantia de 100% de processamento para as máquinas virtuais, com comportamento de usuários definido e cargas de trabalho com múltiplas prioridades. O algoritmo de escalonamento que apresentou melhores resultados (3.138,70 ± 44,81 s) para este quesito foi o `MPD`, empatado tecnicamente pelo menos com o segundo melhor, `RR` (3.207,66 ± 35,17 s).

6.4.10 Análise de Resultados: Tempo para Conclusão de Cargas

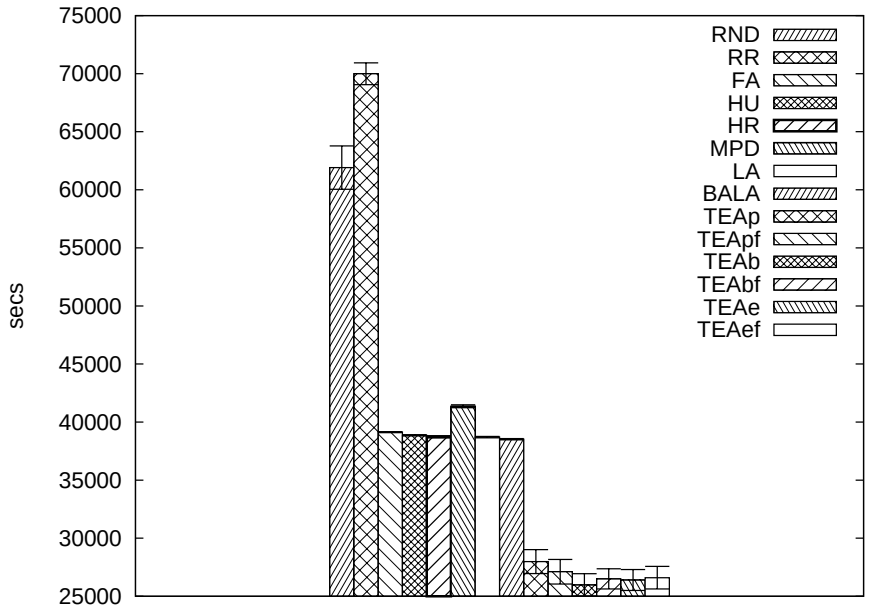

Figura 6.77: Tempo para Conclusão de Cargas: DC Geo. Homo., 100 srvs, K-Ary Fat Tree, Mig, S/SLA, Comp Usr Def, Mult Pr

Apresentamos na Figura 6.77 uma comparação do tempo para conclusão das cargas de trabalho desde o recebimento da solicitação pelo *broker*. Neste cenário, avaliamos uma nuvem possuindo 100 servidores distribuídos em dois *data centers* geodistribuídos e homogêneos, empregando topologia *K-Ary Fat Tree*. O recurso de migração de máquinas virtuais está ativado, sem SLA de garantia de 100% de processamento para as máquinas virtuais, com comportamento de usuários definido e cargas de trabalho com múltiplas prioridades. O algoritmo de escalonamento que apresentou melhores resultados (25.979,71 ± 958,95 s) para este quesito foi o `TEA` (`TEAb`), apresentando, considerando o intervalo de confiança, isoladamente os melhores valores do parâmetro comparado dentre todos os algoritmos para este cenário, com uma melhora variando de 42,8% a 54,2% comparado ao segundo melhor (`BALA`).

Mostramos na Figura 6.78 um gráfico do tempo para conclusão das cargas de trabalho desde o recebimento da solicitação pelo *broker*. Neste cenário, avaliamos uma nuvem possuindo 1.000 servidores distribuídos em dois *data centers* geodistribuídos e homogêneos, empregando topologia *Single-Hop Star*. O recurso de migração de máquinas virtuais está

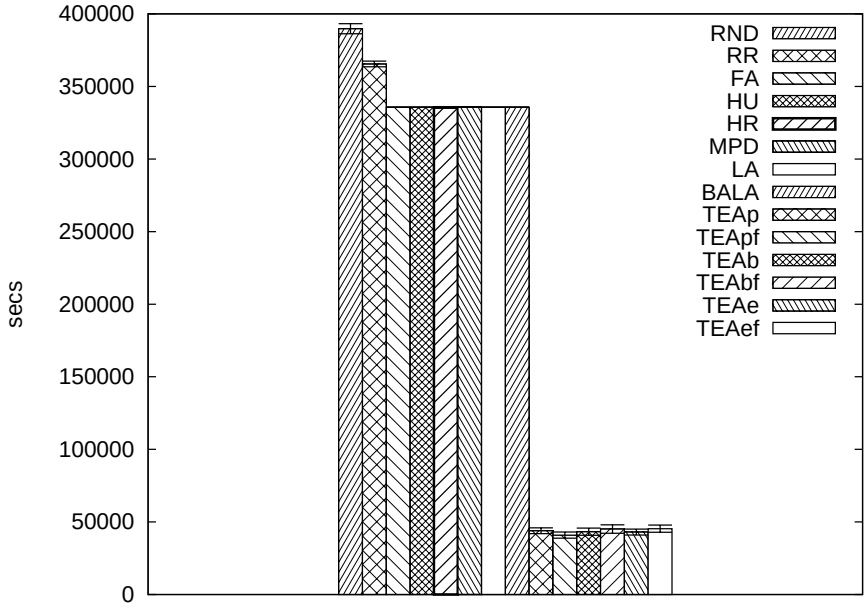

Figura 6.78: Tempo para Conclusão de Cargas: DC Geo. Homo., 1.000 srvs, Single-Hop Star, Mig, SLA, Comp Usr Def, S/Mult Pr

ativado, com SLA de garantia de 100% de processamento para as máquinas virtuais, com comportamento de usuários definido e cargas de trabalho com somente uma prioridade. O algoritmo de escalonamento que apresentou melhores resultados (40.917,59 ± 2.127,68 s) para este quesito foi o `TEA` (`TEApf`), apresentando, considerando o intervalo de confiança, isoladamente os melhores valores do parâmetro comparado dentre todos os algoritmos para este cenário, com uma melhora variando de 679,8% a 765,4% comparado ao segundo melhor (`HU`).

Temos na Figura 6.79 um histograma sobre o tempo para conclusão das cargas de trabalho desde o recebimento da solicitação pelo *broker*. Neste cenário, avaliamos uma nuvem possuindo 10 servidores distribuídos em um *data center* não geodistribuído e heterogêneo, empregando topologia *Binary Tree*. O recurso de migração de máquinas virtuais está desativado, sem SLA de garantia de 100% de processamento para as máquinas virtuais, com comportamento de usuários indefinido e cargas de trabalho com somente uma prioridade. O algoritmo de escalonamento que apresentou melhores resultados (5.601,15 ± 132,10 s) para este quesito foi o `MPD`, empatado tecnicamente pelo menos com o segundo melhor, `BALA` (5.658,52 ± 144,64 s).

Mostramos na Figura 6.80 um gráfico do tempo para conclusão das cargas de trabalho desde o recebimento da solicitação pelo *broker*. Neste cenário, avaliamos uma nuvem possuindo 10 servidores distribuídos em um *data center* não geodistribuído e heterogêneo, empregando topologia *Binary Tree*. O recurso de migração de máquinas virtuais está desativado, com SLA de garantia de 100% de processamento para as máquinas virtuais, com comportamento de usuários indefinido e cargas de trabalho com múltiplas prioridades. O algoritmo de escalonamento que apresentou melhores resultados (5.442,07 ± 14,01 s) para este quesito foi o `RR`, apresentando, considerando o intervalo de confiança, isoladamente os

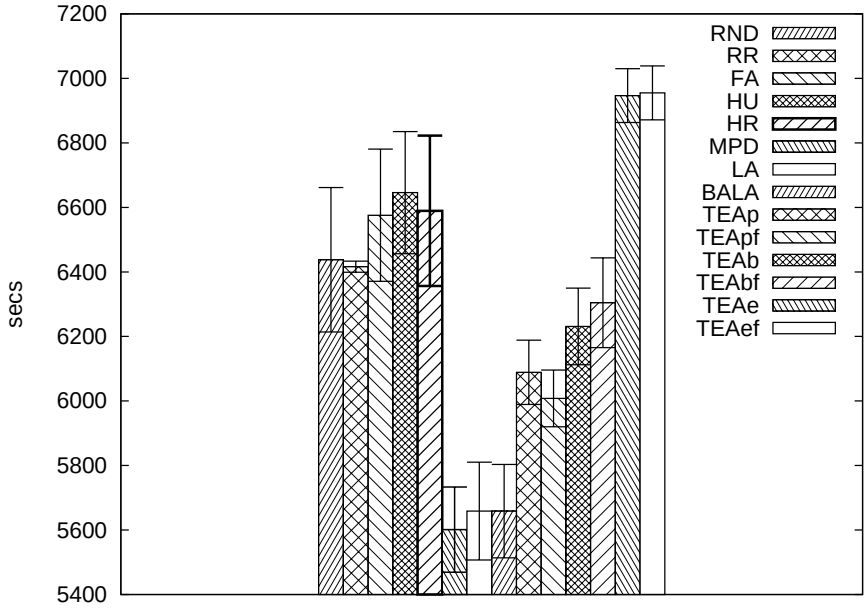

Figura 6.79: Tempo para Conclusão de Cargas: DC N.Geo. Het., 10 srvs, Binary Tree, S/Mig, S/SLA, Comp Usr Ind, S/Mult Pr

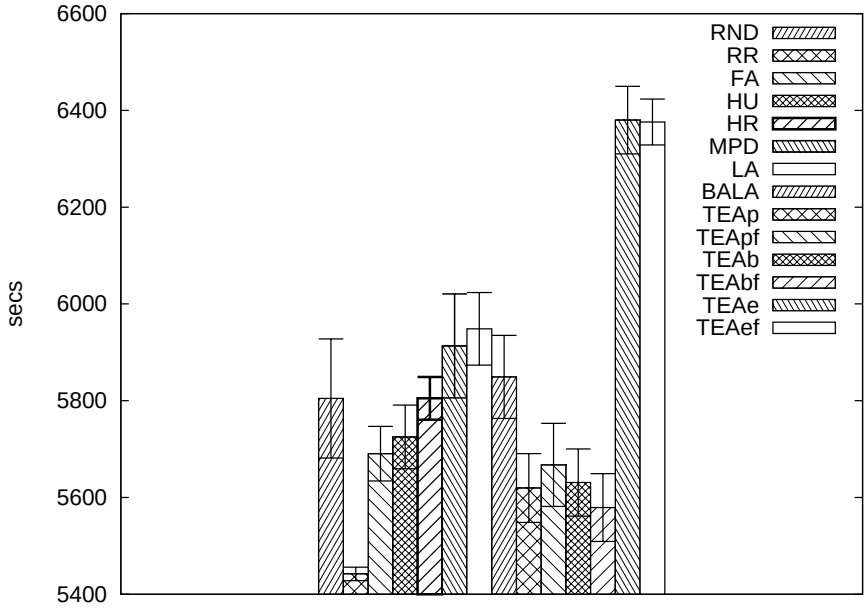

Figura 6.80: Tempo para Conclusão de Cargas: DC N.Geo. Het., 10 srvs, Binary Tree, S/Mig, SLA, Comp Usr Ind, Mult Pr

melhores valores do parâmetro comparado dentre todos os algoritmos para este cenário, com uma melhora variando de 1,0% a 4,1% comparado ao segundo melhor (`TEAbf`).

Temos na Figura 6.81 um histograma sobre o tempo para conclusão das cargas de trabalho desde o recebimento da solicitação pelo *broker*. Neste cenário, avaliamos uma nuvem possuindo 10 servidores distribuídos em um *data center* não geodistribuído e heterogêneo, empregando topologia *Binary Tree*. O recurso de migração de máquinas virtuais está desativado, com SLA de garantia de 100% de processamento para as máquinas virtuais, com

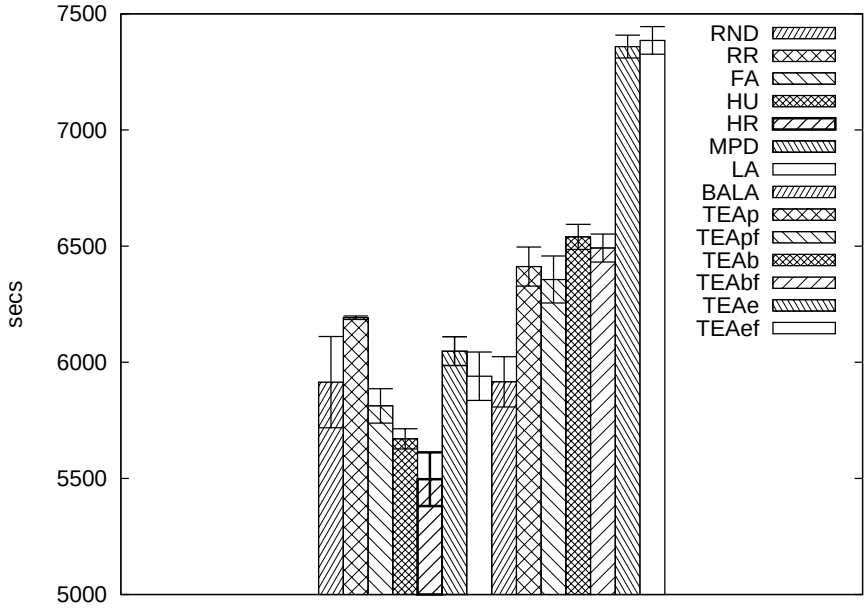

Figura 6.81: Tempo para Conclusão de Cargas: DC N.Geo. Het., 10 srvs, Binary Tree, S/Mig, SLA, Comp Usr Ind, S/Mult Pr

comportamento de usuários indefinido e cargas de trabalho com somente uma prioridade. O algoritmo de escalonamento que apresentou melhores resultados (5.496,62 $\pm$ 115,52 s) para este quesito foi o `HR`, apresentando, considerando o intervalo de confiança, isoladamente os melhores valores do parâmetro comparado dentre todos os algoritmos para este cenário, com uma melhora variando de 0,3% a 6,2% comparado ao segundo melhor (`HU`).

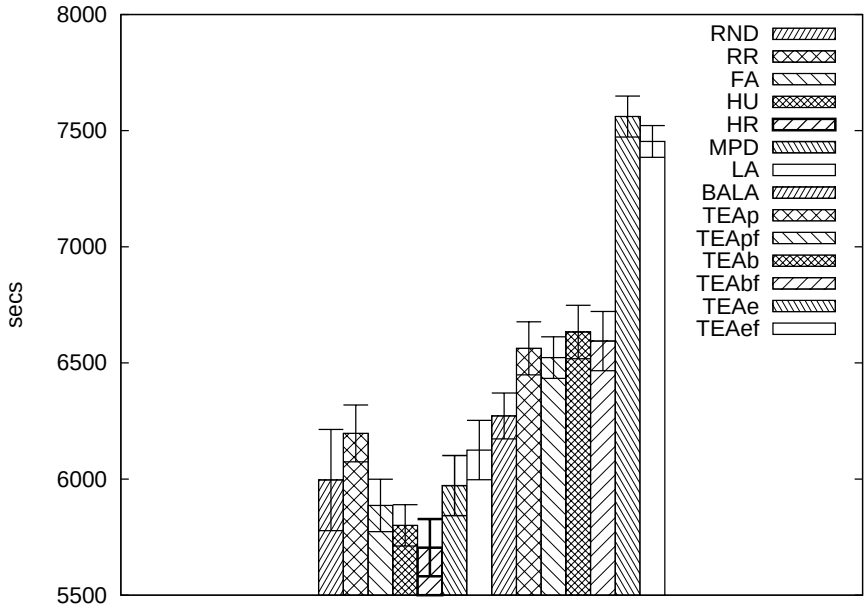

Figura 6.82: Tempo para Conclusão de Cargas: DC N.Geo. Het., 10 srvs, Flat-Tree, Mig, SLA, Comp Usr Ind, S/Mult Pr

Apresentamos na Figura 6.82 uma comparação do tempo para conclusão das cargas de trabalho desde o recebimento da solicitação pelo *broker*. Neste cenário, avaliamos uma nuvem possuindo 10 servidores distribuídos em um *data center* não geodistribuído e heterogêneo, empregando topologia *Flat-Tree*. O recurso de migração de máquinas virtuais está ativado, com SLA de garantia de 100% de processamento para as máquinas virtuais, com comportamento de usuários indefinido e cargas de trabalho com somente uma prioridade. O algoritmo de escalonamento que apresentou melhores resultados (5.704,61 ± 123,29 s) para este quesito foi o `HR`, empatado tecnicamente pelo menos com o segundo melhor, `HU` (5.800,78 ± 89,00 s).

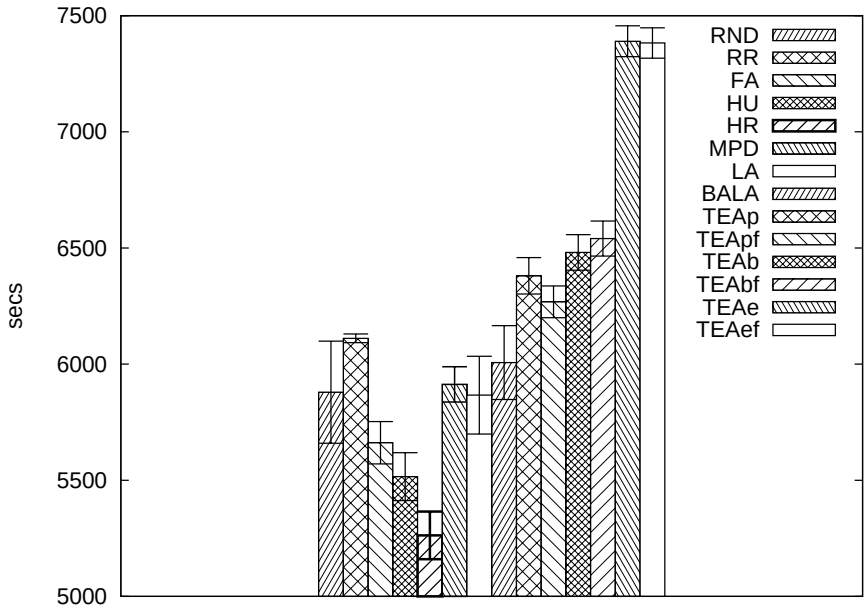

Figura 6.83: Tempo para Conclusão de Cargas: DC N.Geo. Het., 10 srvs, Flat-Tree, S/Mig, SLA, Comp Usr Def, S/Mult Pr

Temos na Figura 6.83 um histograma sobre o tempo para conclusão das cargas de trabalho desde o recebimento da solicitação pelo *broker*. Neste cenário, avaliamos uma nuvem possuindo 10 servidores distribuídos em um *data center* não geodistribuído e heterogêneo, empregando topologia *Flat-Tree*. O recurso de migração de máquinas virtuais está desativado, com SLA de garantia de 100% de processamento para as máquinas virtuais, com comportamento de usuários definido e cargas de trabalho com somente uma prioridade. O algoritmo de escalonamento que apresentou melhores resultados (5.262,78 ± 102,61 s) para este quesito foi o `HR`, apresentando, considerando o intervalo de confiança, isoladamente os melhores valores do parâmetro comparado dentre todos os algoritmos para este cenário, com uma melhora variando de 0,9% a 8,9% comparado ao segundo melhor (`HU`).

Apresentamos na Figura 6.84 uma comparação do tempo para conclusão das cargas de trabalho desde o recebimento da solicitação pelo *broker*. Neste cenário, avaliamos uma nuvem possuindo 10 servidores distribuídos em um *data center* não geodistribuído e heterogêneo, empregando topologia *Flat-Tree*. O recurso de migração de máquinas virtuais está desativado, com SLA de garantia de 100% de processamento para as máquinas virtuais, com comportamento de usuários indefinido e cargas de trabalho com somente uma

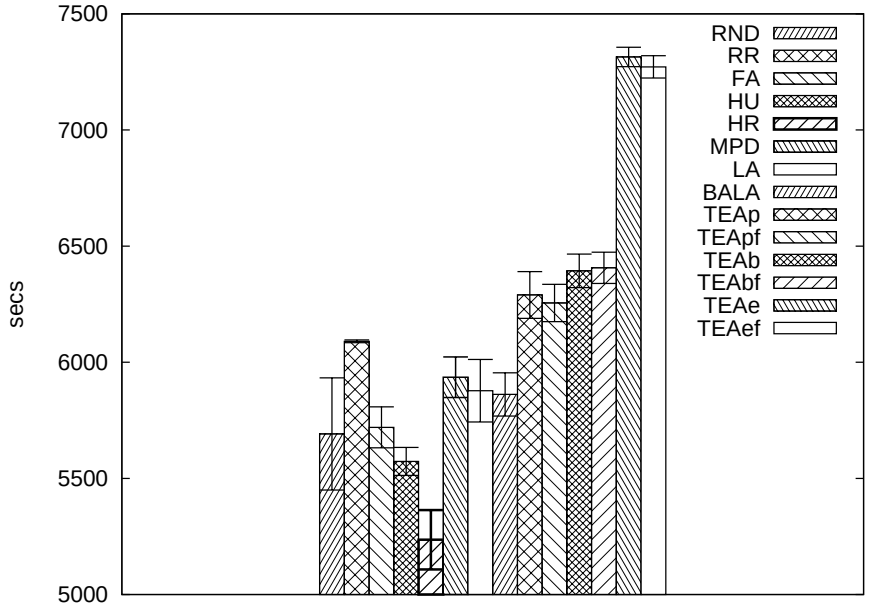

Figura 6.84: Tempo para Conclusão de Cargas: DC N.Geo. Het., 10 srvs, Flat-Tree, S/Mig, SLA, Comp Usr Ind, S/Mult Pr

prioridade. O algoritmo de escalonamento que apresentou melhores resultados (5.235,89 ± 127,72 s) para este quesito foi o `HR`, apresentando, considerando o intervalo de confiança, isoladamente os melhores valores do parâmetro comparado dentre todos os algoritmos para este cenário, com uma melhora variando de 2,8% a 10,3% comparado ao segundo melhor (`HU`).

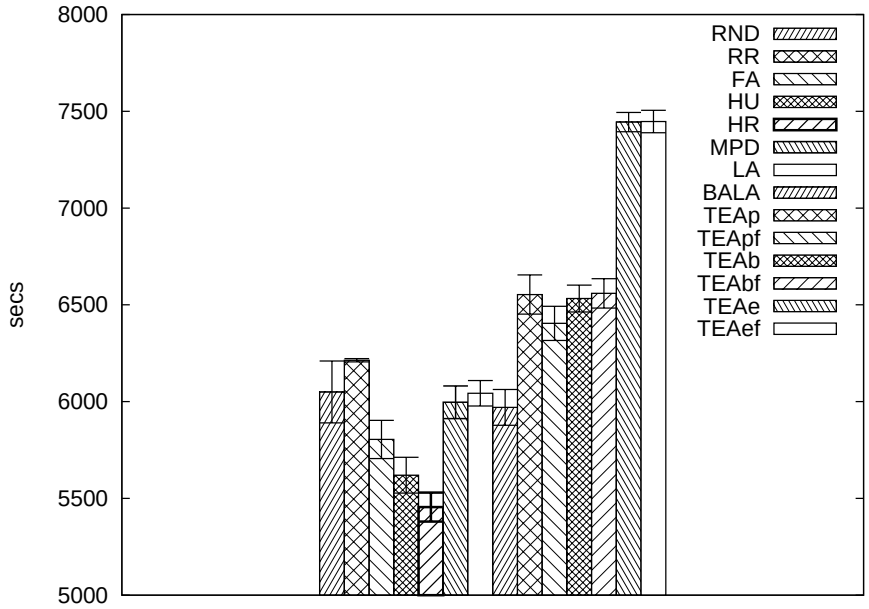

Figura 6.85: Tempo para Conclusão de Cargas: DC N.Geo. Het., 10 srvs, K-Ary Fat Tree, S/Mig, SLA, Comp Usr Ind, S/Mult Pr

Apresentamos na Figura 6.85 uma comparação do tempo para conclusão das cargas de trabalho desde o recebimento da solicitação pelo *broker*. Neste cenário, avaliamos uma nuvem possuindo 10 servidores distribuídos em um *data center* não geodistribuído e heterogêneo, empregando topologia *K-Ary Fat Tree*. O recurso de migração de máquinas virtuais está desativado, com SLA de garantia de 100% de processamento para as máquinas virtuais, com comportamento de usuários indefinido e cargas de trabalho com somente uma prioridade. O algoritmo de escalonamento que apresentou melhores resultados (5.455,18 ± 74,98 s) para este quesito foi o `HR`, empatado tecnicamente pelo menos com o segundo melhor, `HU` (5.619,92 ± 92,42 s).

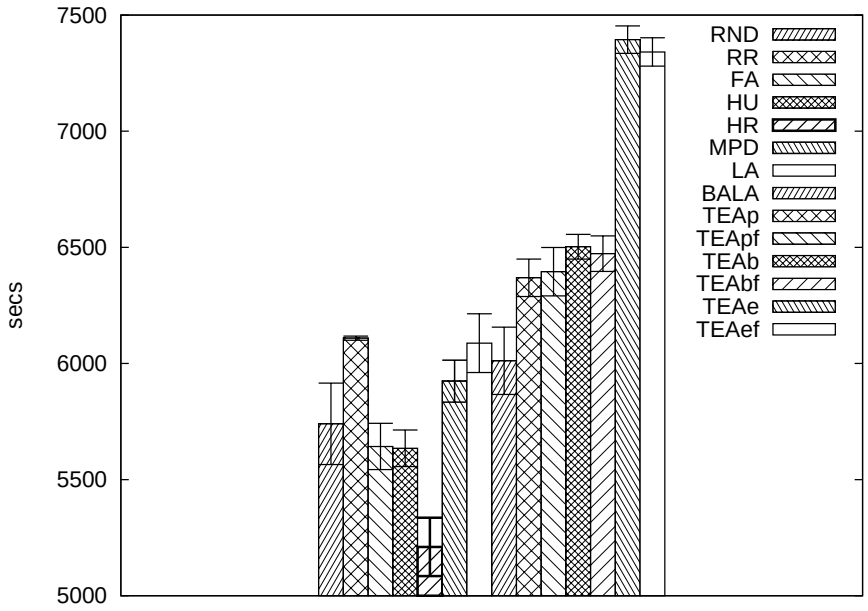

Figura 6.86: Tempo para Conclusão de Cargas: DC N.Geo. Het., 10 srvs, Single-Hop Star, S/Mig, SLA, Comp Usr Def, S/Mult Pr

A Figura 6.86 traz informações sobre o tempo para conclusão das cargas de trabalho desde o recebimento da solicitação pelo *broker*. Neste cenário, avaliamos uma nuvem possuindo 10 servidores distribuídos em um *data center* não geodistribuído e heterogêneo, empregando topologia *Single-Hop Star*. O recurso de migração de máquinas virtuais está desativado, com SLA de garantia de 100% de processamento para as máquinas virtuais, com comportamento de usuários definido e cargas de trabalho com somente uma prioridade. O algoritmo de escalonamento que apresentou melhores resultados (5.210,20 ± 125,69 s) para este quesito foi o `HR`, apresentando, considerando o intervalo de confiança, isoladamente os melhores valores do parâmetro comparado dentre todos os algoritmos para este cenário, com uma melhora variando de 4,1% a 12,4% comparado ao segundo melhor (`HU`).

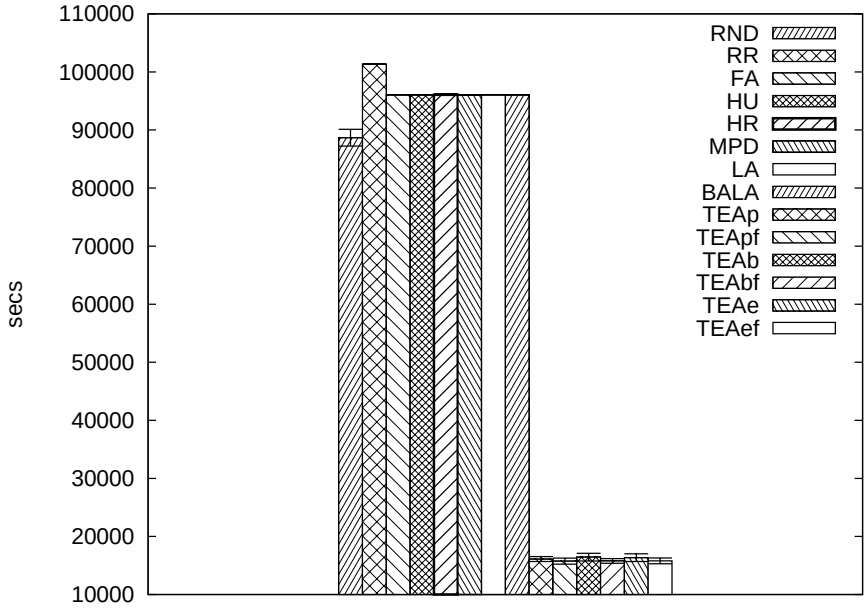

Figura 6.87: Tempo para Conclusão de Cargas: Alta Prioridade: DC Geo. Homo., 1.000 srvs, Binary Tree, S/Mig, SLA, Comp Usr Def, Mult Pr

6.4.11 Análise de Resultados: Tempo para Conclusão de Cargas — Prioridade Alta

Mostramos na Figura 6.87 um gráfico do tempo para conclusão das cargas de trabalho de alta prioridade desde o recebimento da solicitação pelo *broker*. Neste cenário, avaliamos uma nuvem possuindo 1.000 servidores distribuídos em dois *data centers* geodistribuídos e homogêneos, empregando topologia *Binary Tree*. O recurso de migração de máquinas virtuais está desativado, com SLA de garantia de 100% de processamento para as máquinas virtuais, com comportamento de usuários definido e cargas de trabalho com múltiplas prioridades. O algoritmo de escalonamento que apresentou melhores resultados (15.759,20 ± 519,82 s) para este quesito foi o `TEA` (`TEApf`), apresentando, considerando o intervalo de confiança, isoladamente os melhores valores do parâmetro comparado dentre todos os algoritmos para este cenário, com uma melhora variando de 435,9% a 491,3% comparado ao segundo melhor (`RND`).

Mostramos na Figura 6.88 um gráfico do tempo para conclusão das cargas de trabalho de alta prioridade desde o recebimento da solicitação pelo *broker*. Neste cenário, avaliamos uma nuvem possuindo 1.000 servidores distribuídos em dois *data centers* geodistribuídos e heterogêneos, empregando topologia *Flat-Tree*. O recurso de migração de máquinas virtuais está desativado, sem SLA de garantia de 100% de processamento para as máquinas virtuais, com comportamento de usuários definido e cargas de trabalho com múltiplas prioridades. O algoritmo de escalonamento que apresentou melhores resultados (28.011,15 ± 685,40 s) para este quesito foi o `TEA` (`TEApf`), apresentando, considerando o intervalo de confiança, isoladamente os melhores valores do parâmetro comparado dentre todos os algoritmos para este cenário, com uma melhora variando de 201,2% a 228,3% comparado ao segundo melhor (`RND`).

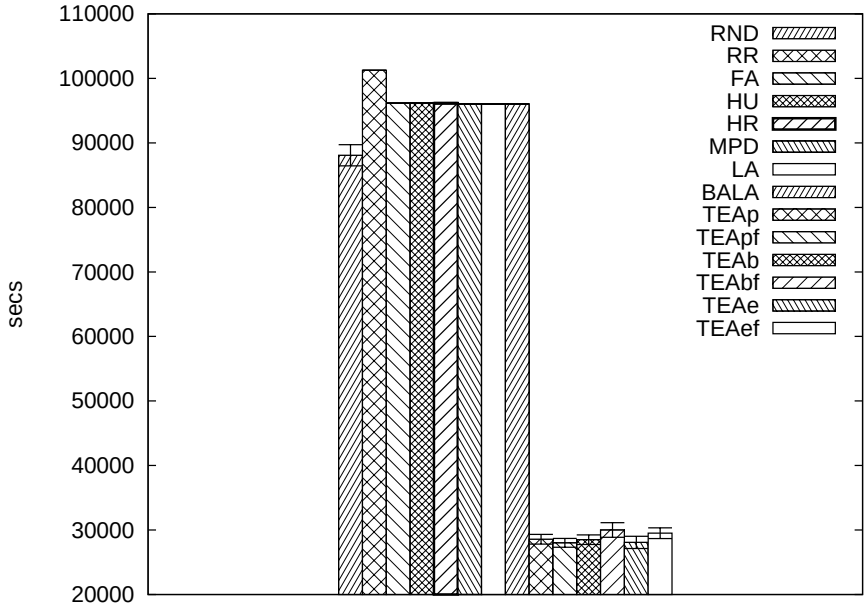

Figura 6.88: Tempo para Conclusão de Cargas: Alta Prioridade: DC Geo. Het., 1.000 srvs, Flat-Tree, S/Mig, S/SLA, Comp Usr Def, Mult Pr

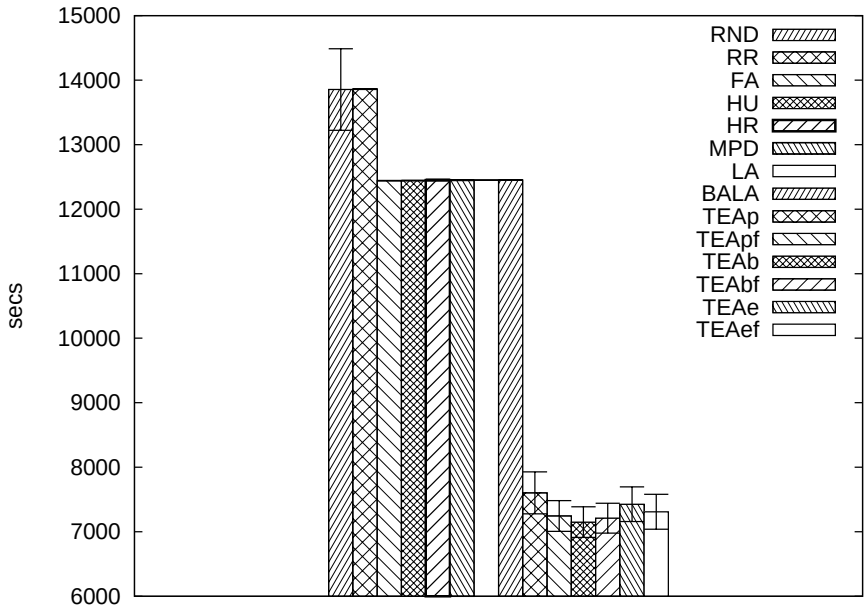

Figura 6.89: Tempo para Conclusão de Cargas: Alta Prioridade: DC Geo. Homo., 100 srvs, Flat-Tree, S/Mig, SLA, Comp Usr Ind, Mult Pr

Mostramos na Figura 6.89 um gráfico do tempo para conclusão das cargas de trabalho de alta prioridade desde o recebimento da solicitação pelo *broker*. Neste cenário, avaliamos uma nuvem possuindo 100 servidores distribuídos em dois *data centers* geodistribuídos e homogêneos, empregando topologia *Flat-Tree*. O recurso de migração de máquinas virtuais está desativado, com SLA de garantia de 100% de processamento para as máquinas virtuais, com comportamento de usuários indefinido e cargas de trabalho com múltiplas prioridades. O algoritmo de escalonamento que apresentou melhores resultados (7.149,36

± 236,80 s) para este quesito foi o `TEA` (`TEAb`), apresentando, considerando o intervalo de confiança, isoladamente os melhores valores do parâmetro comparado dentre todos os algoritmos para este cenário, com uma melhora variando de 68,4% a 80,0% comparado ao segundo melhor (`FA`).

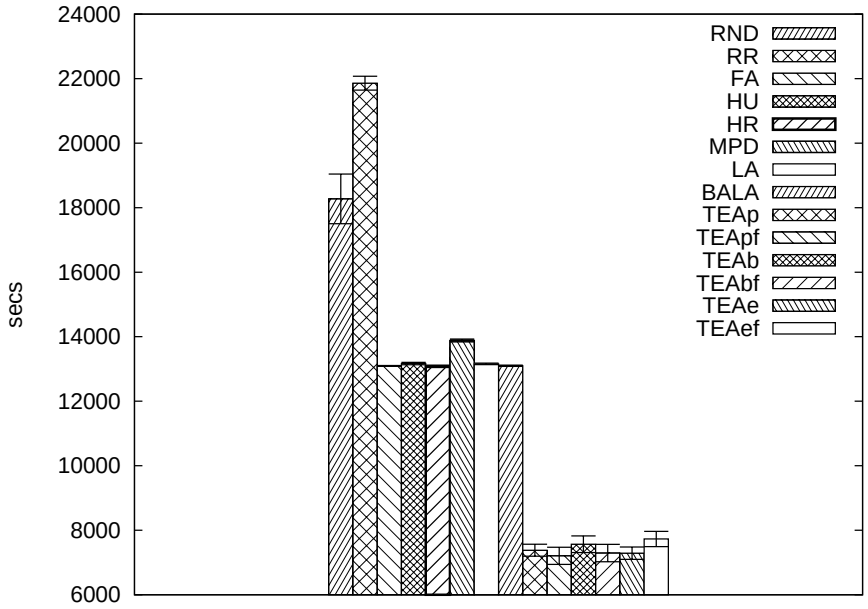

Figura 6.90: Tempo para Conclusão de Cargas: Alta Prioridade: DC Geo. Homo., 100 srvs, K-Ary Fat Tree, Mig, S/SLA, Comp Usr Ind, Mult Pr

Apresentamos na Figura 6.90 uma comparação do tempo para conclusão das cargas de trabalho de alta prioridade desde o recebimento da solicitação pelo *broker*. Neste cenário, avaliamos uma nuvem possuindo 100 servidores distribuídos em dois *data centers* geodistribuídos e homogêneos, empregando topologia *K-Ary Fat Tree*. O recurso de migração de máquinas virtuais está ativado, sem SLA de garantia de 100% de processamento para as máquinas virtuais, com comportamento de usuários indefinido e cargas de trabalho com múltiplas prioridades. O algoritmo de escalonamento que apresentou melhores resultados (7.207,13 ± 264,17 s) para este quesito foi o `TEA` (`TEApf`), apresentando, considerando o intervalo de confiança, isoladamente os melhores valores do parâmetro comparado dentre todos os algoritmos para este cenário, com uma melhora variando de 74,8% a 88,7% comparado ao segundo melhor (`HR`).

A Figura 6.91 traz informações sobre o tempo para conclusão das cargas de trabalho de alta prioridade desde o recebimento da solicitação pelo *broker*. Neste cenário, avaliamos uma nuvem possuindo 100 servidores distribuídos em dois *data centers* geodistribuídos e homogêneos, empregando topologia *Single-Hop Star*. O recurso de migração de máquinas virtuais está ativado, sem SLA de garantia de 100% de processamento para as máquinas virtuais, com comportamento de usuários indefinido e cargas de trabalho com múltiplas prioridades. O algoritmo de escalonamento que apresentou melhores resultados (6.578,57 ± 196,09 s) para este quesito foi o `TEA` (`TEAbf`), apresentando, considerando o intervalo de confiança, isoladamente os melhores valores do parâmetro comparado dentre todos os

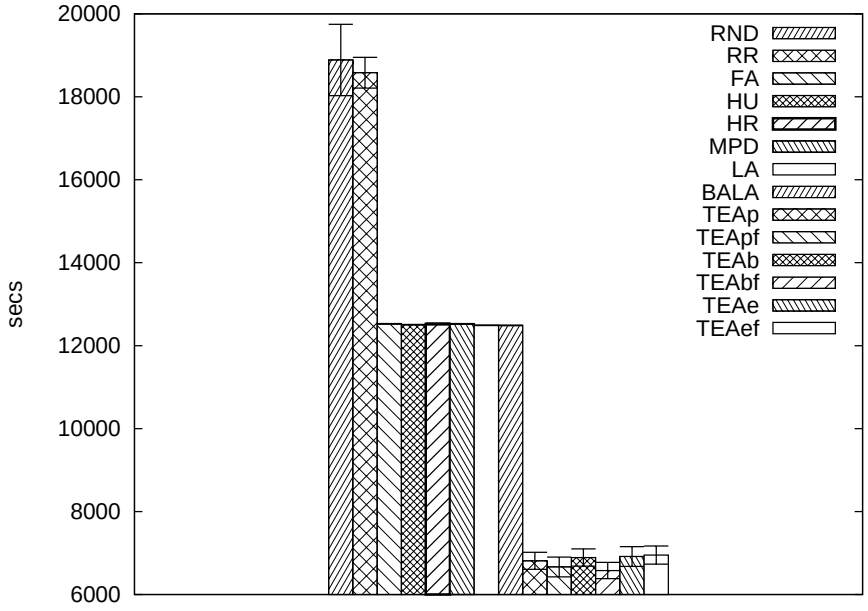

Figura 6.91: Tempo para Conclusão de Cargas: Alta Prioridade: DC Geo. Homo., 100 srvs, Single-Hop Star, Mig, S/SLA, Comp Usr Ind, Mult Pr

algoritmos para este cenário, com uma melhora variando de 84,3% a 95,8% comparado ao segundo melhor (`BALA`).

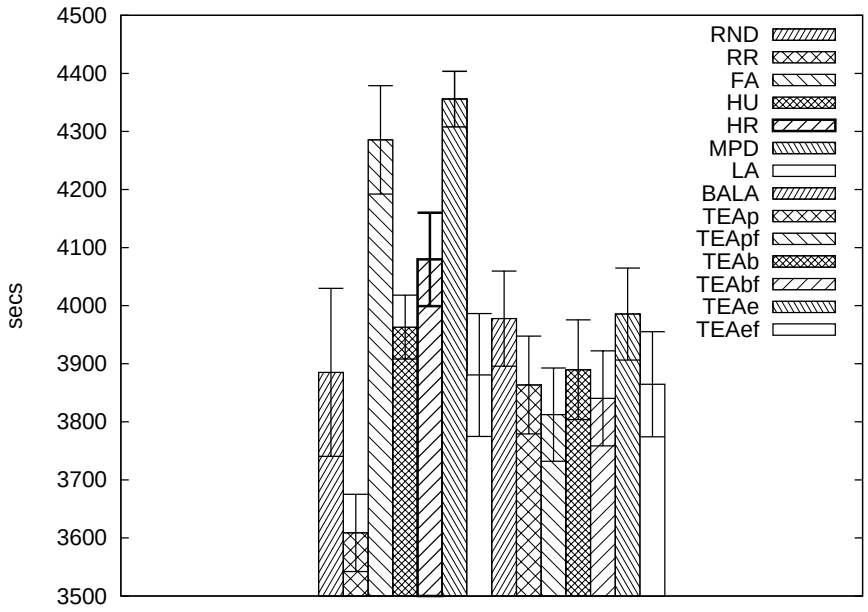

Figura 6.92: Tempo para Conclusão de Cargas: Alta Prioridade: DC N.Geo. Het., 10 srvs, Single-Hop Star, Mig, S/SLA, Comp Usr Def, Mult Pr

A Figura 6.92 traz informações sobre o tempo para conclusão das cargas de trabalho de alta prioridade desde o recebimento da solicitação pelo *broker*. Neste cenário, avaliamos uma nuvem possuindo 10 servidores distribuídos em um *data center* não geodistribuído e heterogêneo, empregando topologia *Single-Hop Star*. O recurso de migração de máquinas

virtuais está ativado, sem SLA de garantia de 100% de processamento para as máquinas virtuais, com comportamento de usuários definido e cargas de trabalho com múltiplas prioridades. O algoritmo de escalonamento que apresentou melhores resultados (3.608,69 ± 66,51 s) para este quesito foi o `RR`, apresentando, considerando o intervalo de confiança, isoladamente os melhores valores do parâmetro comparado dentre todos os algoritmos para este cenário, com uma melhora variando de 1,5% a 9,9% comparado ao segundo melhor (`TEApf`).

6.4.12 Análise de Resultados: Tempo para Conclusão de Cargas — Prioridade Normal

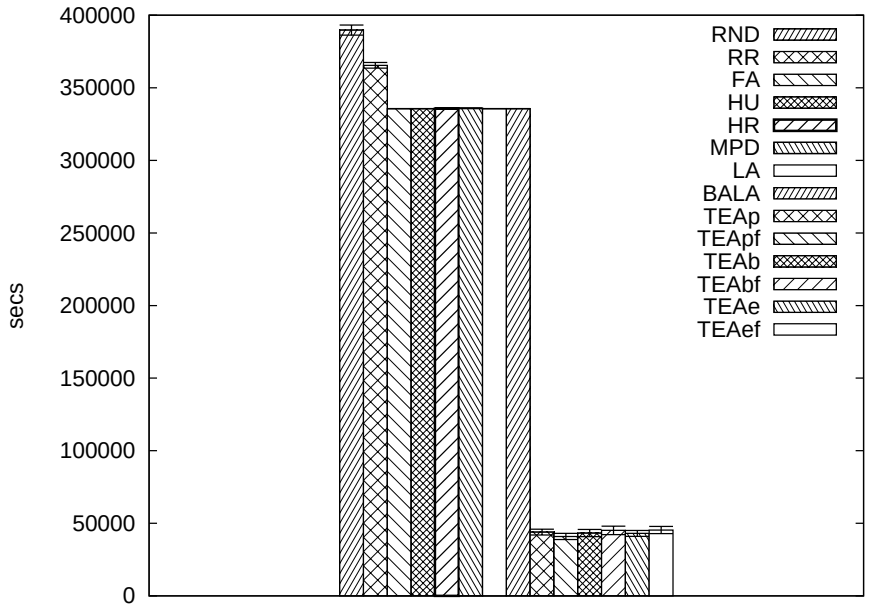

Figura 6.93: Tempo para Conclusão de Cargas: Prioridade Normal: DC Geo. Homo., 1.000 srvs, Single-Hop Star, Mig, SLA, Comp Usr Def, S/Mult Pr

Mostramos na Figura 6.93 um gráfico do tempo para conclusão das cargas de trabalho de prioridade normal desde o recebimento da solicitação pelo *broker*. Neste cenário, avaliamos uma nuvem possuindo 1.000 servidores distribuídos em dois *data centers* geodistribuídos e homogêneos, empregando topologia *Single-Hop Star*. O recurso de migração de máquinas virtuais está ativado, com SLA de garantia de 100% de processamento para as máquinas virtuais, com comportamento de usuários definido e cargas de trabalho com somente uma prioridade. O algoritmo de escalonamento que apresentou melhores resultados (40.917,59 ± 2.127,68 s) para este quesito foi o `TEA` (`TEApf`), apresentando, considerando o intervalo de confiança, isoladamente os melhores valores do parâmetro comparado dentre todos os algoritmos para este cenário, com uma melhora variando de 679,8% a 765,4% comparado ao segundo melhor (`HU`).

A Figura 6.94 traz informações sobre o tempo para conclusão das cargas de trabalho de prioridade normal desde o recebimento da solicitação pelo *broker*. Neste cenário, avaliamos uma nuvem possuindo 100 servidores distribuídos em dois *data centers* geodistribuídos e

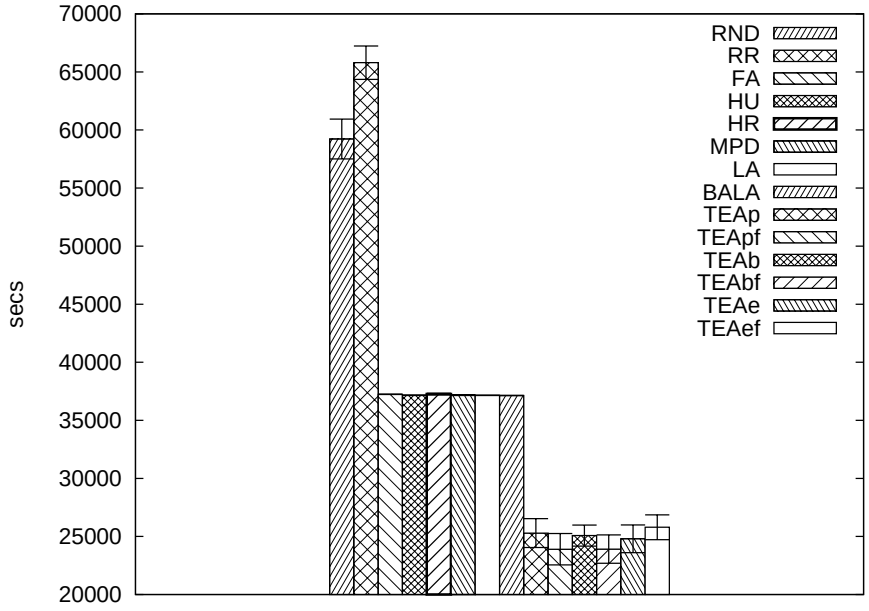

Figura 6.94: Tempo para Conclusão de Cargas: Prioridade Normal: DC Geo. Homo., 100 srvs, Single-Hop Star, Mig, S/SLA, Comp Usr Ind, Mult Pr

homogêneos, empregando topologia *Single-Hop Star*. O recurso de migração de máquinas virtuais está ativado, sem SLA de garantia de 100% de processamento para as máquinas virtuais, com comportamento de usuários indefinido e cargas de trabalho com múltiplas prioridades. O algoritmo de escalonamento que apresentou melhores resultados (23.901,10 ± 1.354,26 s) para este quesito foi o `TEA` (`TEApf`), apresentando, considerando o intervalo de confiança, isoladamente os melhores valores do parâmetro comparado dentre todos os algoritmos para este cenário, com uma melhora variando de 47,0% a 64,8% comparado ao segundo melhor (`BALA`).

Apresentamos na Figura 6.95 uma comparação do tempo para conclusão das cargas de trabalho de prioridade normal desde o recebimento da solicitação pelo *broker*. Neste cenário, avaliamos uma nuvem possuindo 10 servidores distribuídos em um *data center* não geodistribuído e heterogêneo, empregando topologia *Binary Tree*. O recurso de migração de máquinas virtuais está desativado, sem SLA de garantia de 100% de processamento para as máquinas virtuais, com comportamento de usuários indefinido e cargas de trabalho com somente uma prioridade. O algoritmo de escalonamento que apresentou melhores resultados (5.601,15 ± 132,10 s) para este quesito foi o `MPD`, empatado tecnicamente pelo menos com o segundo melhor, `BALA` (5.658,52 ± 144,64 s).

Temos na Figura 6.96 um histograma sobre o tempo para conclusão das cargas de trabalho de prioridade normal desde o recebimento da solicitação pelo *broker*. Neste cenário, avaliamos uma nuvem possuindo 10 servidores distribuídos em um *data center* não geodistribuído e heterogêneo, empregando topologia *Binary Tree*. O recurso de migração de máquinas virtuais está desativado, com SLA de garantia de 100% de processamento para as máquinas virtuais, com comportamento de usuários indefinido e cargas de trabalho com somente uma prioridade. O algoritmo de escalonamento que apresentou melhores resultados (5.496,62 ± 115,52 s) para este quesito foi o `HR`, apresentando, considerando

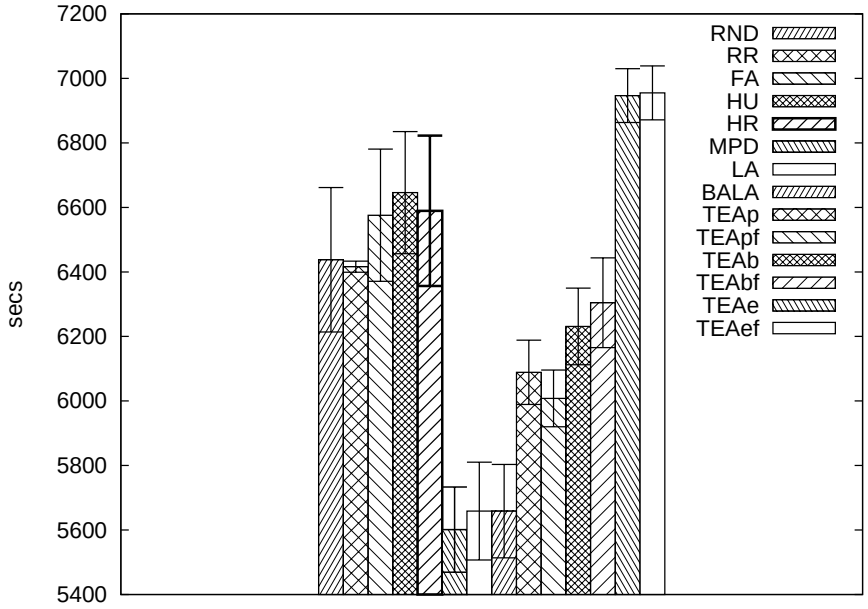

Figura 6.95: Tempo para Conclusão de Cargas: Prioridade Normal: DC N.Geo. Het., 10 srvs, Binary Tree, S/Mig, S/SLA, Comp Usr Ind, S/Mult Pr

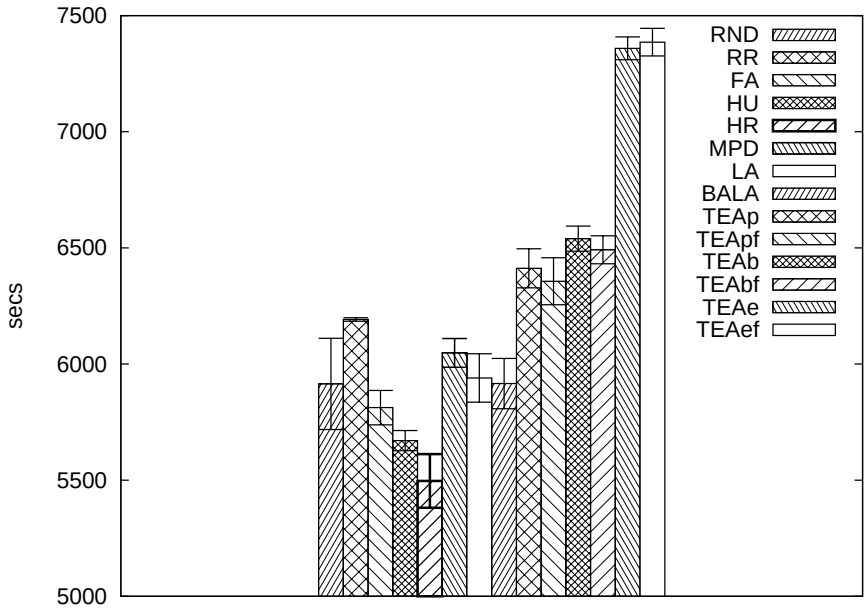

Figura 6.96: Tempo para Conclusão de Cargas: Prioridade Normal: DC N.Geo. Het., 10 srvs, Binary Tree, S/Mig, SLA, Comp Usr Ind, S/Mult Pr

o intervalo de confiança, isoladamente os melhores valores do parâmetro comparado dentre todos os algoritmos para este cenário, com uma melhora variando de 0,3% a 6,2% comparado ao segundo melhor (`HU`).

Mostramos na Figura 6.97 um gráfico do tempo para conclusão das cargas de trabalho de prioridade normal desde o recebimento da solicitação pelo *broker*. Neste cenário, avaliamos uma nuvem possuindo 10 servidores distribuídos em um *data center* não geodistribuído e heterogêneo, empregando topologia *Flat-Tree*. O recurso de migração de

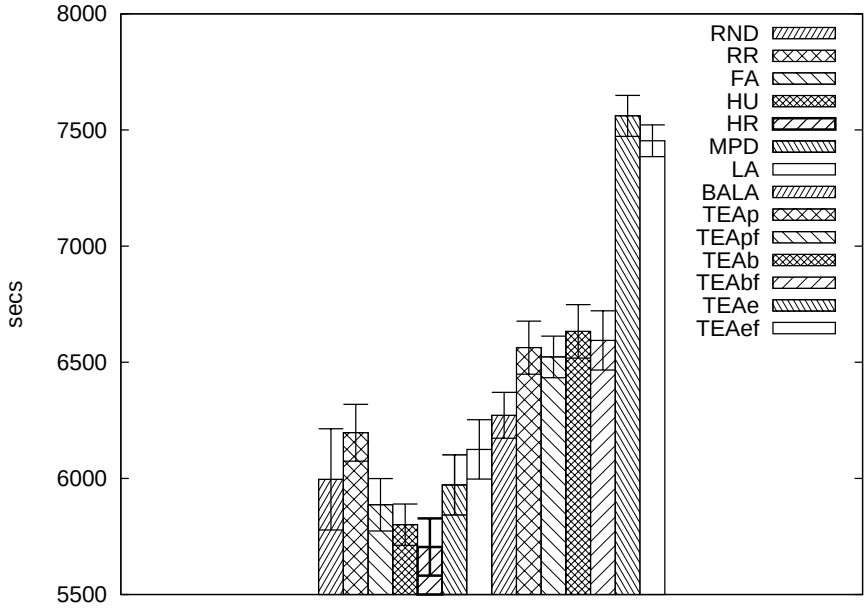

Figura 6.97: Tempo para Conclusão de Cargas: Prioridade Normal: DC N.Geo. Het., 10 srvs, Flat-Tree, Mig, SLA, Comp Usr Ind, S/Mult Pr

máquinas virtuais está ativado, com SLA de garantia de 100% de processamento para as máquinas virtuais, com comportamento de usuários indefinido e cargas de trabalho com somente uma prioridade. O algoritmo de escalonamento que apresentou melhores resultados (5.704,61 ± 123,29 s) para este quesito foi o `HR`, empatado tecnicamente pelo menos com o segundo melhor, `HU` (5.800,78 ± 89,00 s).

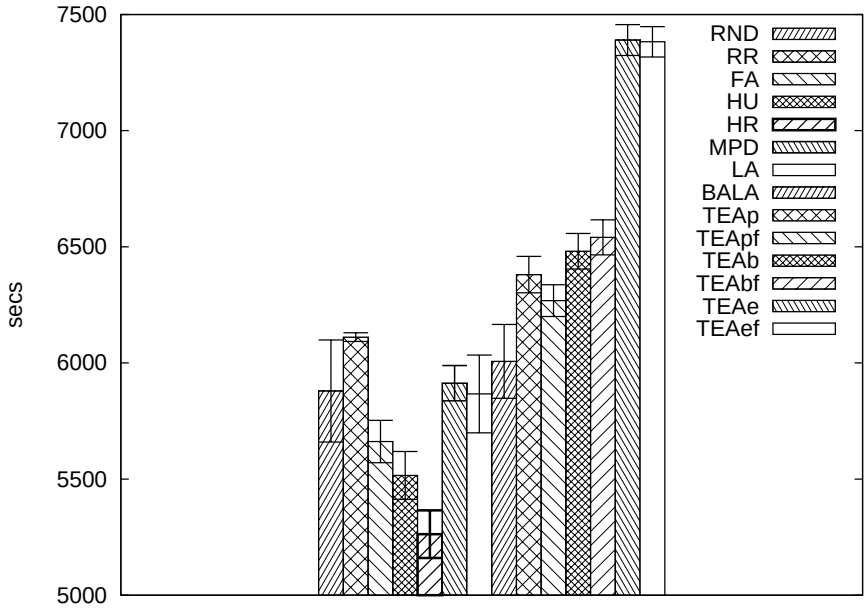

Figura 6.98: Tempo para Conclusão de Cargas: Prioridade Normal: DC N.Geo. Het., 10 srvs, Flat-Tree, S/Mig, SLA, Comp Usr Def, S/Mult Pr

Apresentamos na Figura 6.98 uma comparação do tempo para conclusão das cargas de trabalho de prioridade normal desde o recebimento da solicitação pelo *broker*. Neste cenário, avaliamos uma nuvem possuindo 10 servidores distribuídos em um *data center* não geodistribuído e heterogêneo, empregando topologia *Flat-Tree*. O recurso de migração de máquinas virtuais está desativado, com SLA de garantia de 100% de processamento para as máquinas virtuais, com comportamento de usuários definido e cargas de trabalho com somente uma prioridade. O algoritmo de escalonamento que apresentou melhores resultados (5.262,78 ± 102,61 s) para este quesito foi o `HR`, apresentando, considerando o intervalo de confiança, isoladamente os melhores valores do parâmetro comparado dentre todos os algoritmos para este cenário, com uma melhora variando de 0,9% a 8,9% comparado ao segundo melhor (`HU`).

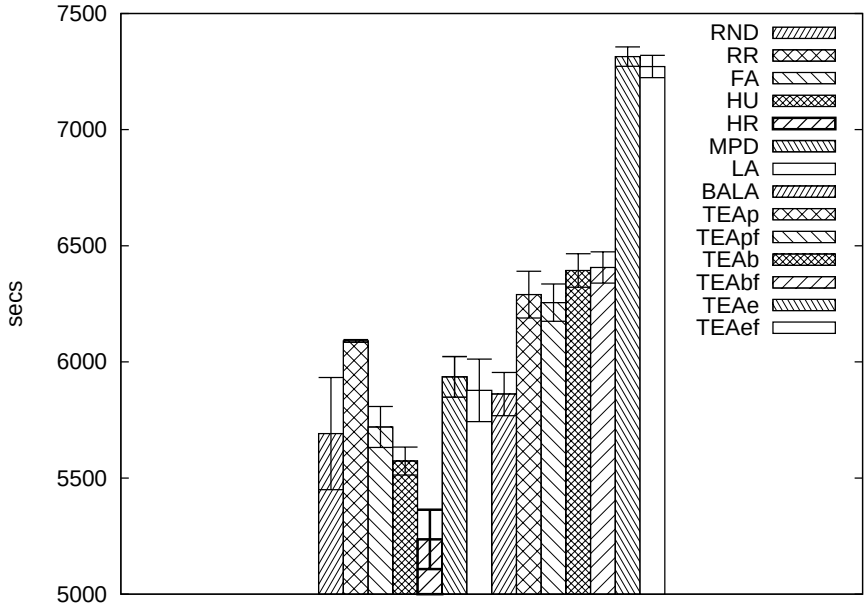

Figura 6.99: Tempo para Conclusão de Cargas: Prioridade Normal: DC N.Geo. Het., 10 srvs, Flat-Tree, S/Mig, SLA, Comp Usr Ind, S/Mult Pr

Apresentamos na Figura 6.99 uma comparação do tempo para conclusão das cargas de trabalho de prioridade normal desde o recebimento da solicitação pelo *broker*. Neste cenário, avaliamos uma nuvem possuindo 10 servidores distribuídos em um *data center* não geodistribuído e heterogêneo, empregando topologia *Flat-Tree*. O recurso de migração de máquinas virtuais está desativado, com SLA de garantia de 100% de processamento para as máquinas virtuais, com comportamento de usuários indefinido e cargas de trabalho com somente uma prioridade. O algoritmo de escalonamento que apresentou melhores resultados (5.235,89 ± 127,72 s) para este quesito foi o `HR`, apresentando, considerando o intervalo de confiança, isoladamente os melhores valores do parâmetro comparado dentre todos os algoritmos para este cenário, com uma melhora variando de 2,8% a 10,3% comparado ao segundo melhor (`HU`).

Apresentamos na Figura 6.100 uma comparação do tempo para conclusão das cargas de trabalho de prioridade normal desde o recebimento da solicitação pelo *broker*. Neste cenário, avaliamos uma nuvem possuindo 10 servidores distribuídos em um *data center*

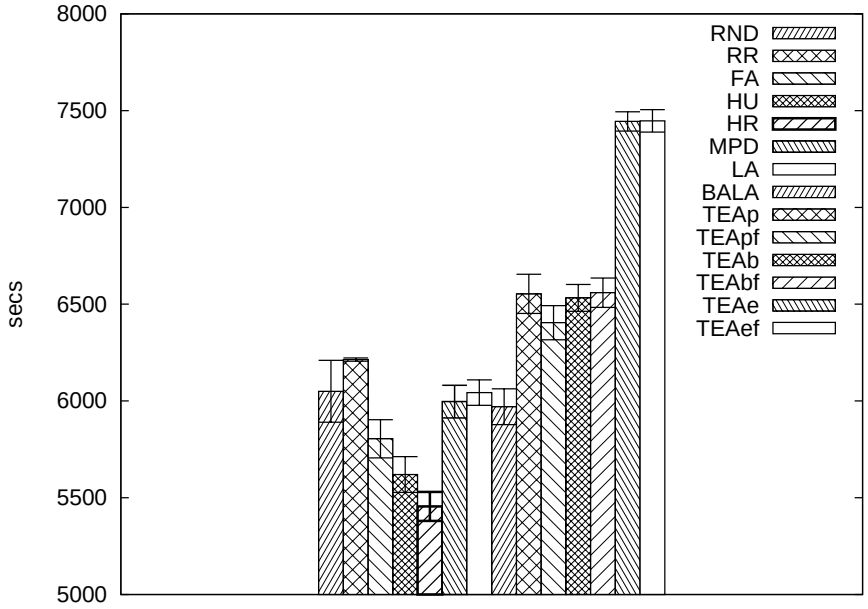

Figura 6.100: Tempo para Conclusão de Cargas: Prioridade Normal: DC N.Geo. Het., 10 srvs, K-Ary Fat Tree, S/Mig, SLA, Comp Usr Ind, S/Mult Pr

não geodistribuído e heterogêneo, empregando topologia *K-Ary Fat Tree*. O recurso de migração de máquinas virtuais está desativado, com SLA de garantia de 100% de processamento para as máquinas virtuais, com comportamento de usuários indefinido e cargas de trabalho com somente uma prioridade. O algoritmo de escalonamento que apresentou melhores resultados (5.455,18 ± 74,98 s) para este quesito foi o `HR`, empatado tecnicamente pelo menos com o segundo melhor, `HU` (5.619,92 ± 92,42 s).

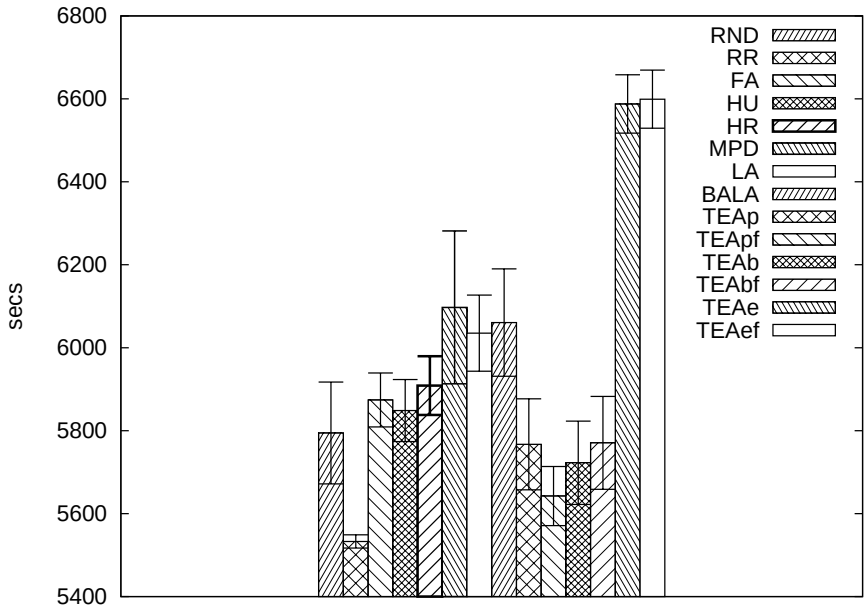

Figura 6.101: Tempo para Conclusão de Cargas: Prioridade Normal: DC N.Geo. Het., 10 srvs, Single-Hop Star, S/Mig, SLA, Comp Usr Ind, Mult Pr

Apresentamos na Figura 6.101 uma comparação do tempo para conclusão das cargas de trabalho de prioridade normal desde o recebimento da solicitação pelo *broker*. Neste cenário, avaliamos uma nuvem possuindo 10 servidores distribuídos em um *data center* não geodistribuído e heterogêneo, empregando topologia *Single-Hop Star*. O recurso de migração de máquinas virtuais está desativado, com SLA de garantia de 100% de processamento para as máquinas virtuais, com comportamento de usuários indefinido e cargas de trabalho com múltiplas prioridades. O algoritmo de escalonamento que apresentou melhores resultados (5.532,93 ± 16,06 s) para este quesito foi o `RR`, apresentando, considerando o intervalo de confiança, isoladamente os melhores valores do parâmetro comparado dentre todos os algoritmos para este cenário, com uma melhora variando de 0,4% a 3,6% comparado ao segundo melhor (`TEApf`).

6.4.13 Análise de Resultados: Tempo para Conclusão de Cargas — Prioridade Baixa

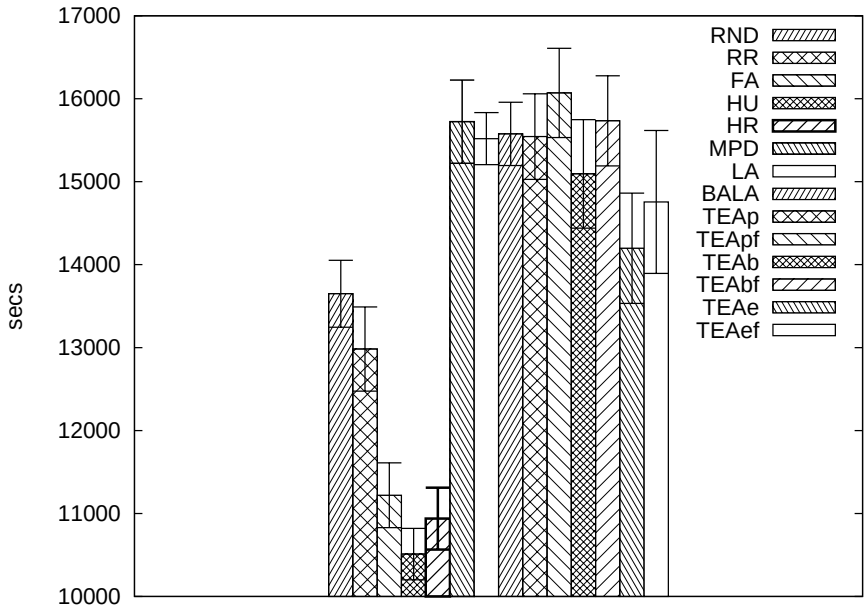

Figura 6.102: Tempo para Conclusão de Cargas: Baixa Prioridade: DC Geo. Het., 10 srvs, K-Ary Fat Tree, Mig, S/SLA, Comp Usr Def, Mult Pr

A Figura 6.102 traz informações sobre o tempo para conclusão das cargas de trabalho de baixa prioridade desde o recebimento da solicitação pelo *broker*. Neste cenário, avaliamos uma nuvem possuindo 10 servidores distribuídos em dois *data centers* geodistribuídos e heterogêneos, empregando topologia *K-Ary Fat Tree*. O recurso de migração de máquinas virtuais está ativado, sem SLA de garantia de 100% de processamento para as máquinas virtuais, com comportamento de usuários definido e cargas de trabalho com múltiplas prioridades. O algoritmo de escalonamento que apresentou melhores resultados (10.510,61 ± 309,75 s) para este quesito foi o `HU`, empatado tecnicamente pelo menos com o segundo melhor, `HR` (10.938,50 ± 372,56 s).

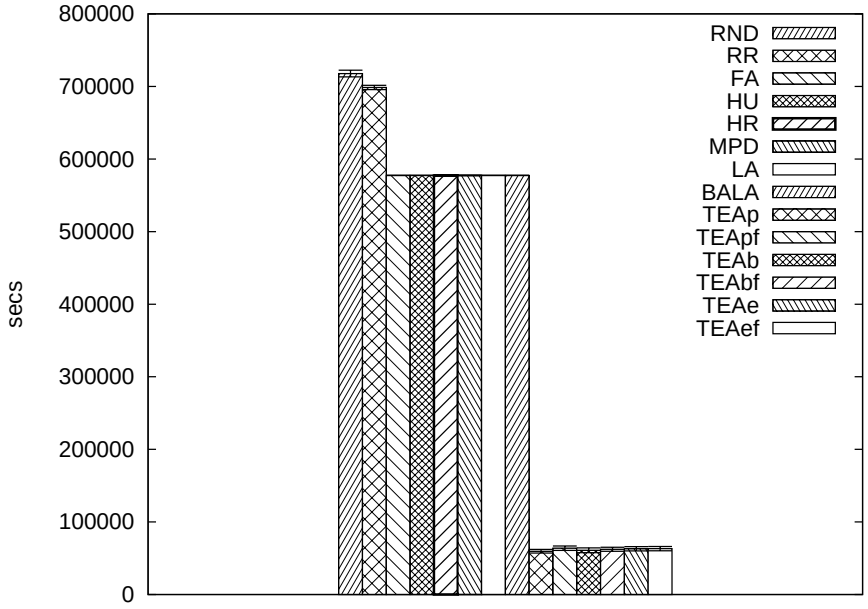

Figura 6.103: Tempo para Conclusão de Cargas: Baixa Prioridade: DC Geo. Homo., 1.000 srvs, Single-Hop Star, Mig, SLA, Comp Usr Def, Mult Pr

A Figura 6.103 traz informações sobre o tempo para conclusão das cargas de trabalho de baixa prioridade desde o recebimento da solicitação pelo *broker*. Neste cenário, avaliamos uma nuvem possuindo 1.000 servidores distribuídos em dois *data centers* geodistribuídos e homogêneos, empregando topologia *Single-Hop Star*. O recurso de migração de máquinas virtuais está ativado, com SLA de garantia de 100% de processamento para as máquinas virtuais, com comportamento de usuários definido e cargas de trabalho com múltiplas prioridades. O algoritmo de escalonamento que apresentou melhores resultados (59.641,89 ± 2.521,64 s) para este quesito foi o `TEA` (`TEAp`), apresentando, considerando o intervalo de confiança, isoladamente os melhores valores do parâmetro comparado dentre todos os algoritmos para este cenário, com uma melhora variando de 828,7% a 910,8% comparado ao segundo melhor (`LA`).

Temos na Figura 6.104 um histograma sobre o tempo para conclusão das cargas de trabalho de baixa prioridade desde o recebimento da solicitação pelo *broker*. Neste cenário, avaliamos uma nuvem possuindo 100 servidores distribuídos em um *data center* não geodistribuído e heterogêneo, empregando topologia *Binary Tree*. O recurso de migração de máquinas virtuais está ativado, sem SLA de garantia de 100% de processamento para as máquinas virtuais, com comportamento de usuários indefinido e cargas de trabalho com múltiplas prioridades. O algoritmo de escalonamento que apresentou melhores resultados (22.071,54 ± 255,46 s) para este quesito foi o `BALA`, empatado tecnicamente pelo menos com o segundo melhor, `LA` (22.328,23 ± 215,50 s).

Mostramos na Figura 6.105 um gráfico do tempo para conclusão das cargas de trabalho de baixa prioridade desde o recebimento da solicitação pelo *broker*. Neste cenário, avaliamos uma nuvem possuindo 10 servidores distribuídos em um *data center* não geodistribuído e heterogêneo, empregando topologia *Binary Tree*. O recurso de migração de máquinas virtuais está ativado, com SLA de garantia de 100% de processamento para

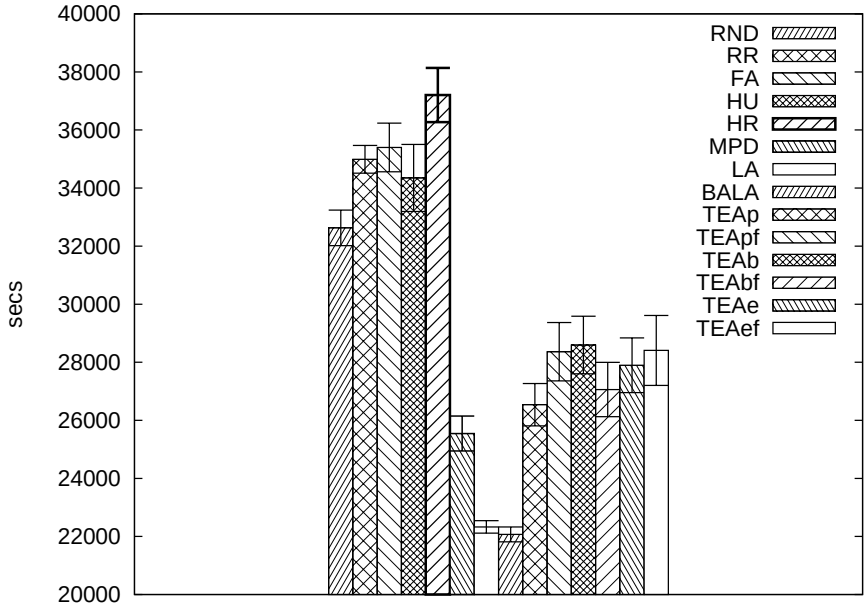

Figura 6.104: Tempo para Conclusão de Cargas: Baixa Prioridade: DC N.Geo. Het., 100 srvs, Binary Tree, Mig, S/SLA, Comp Usr Ind, Mult Pr

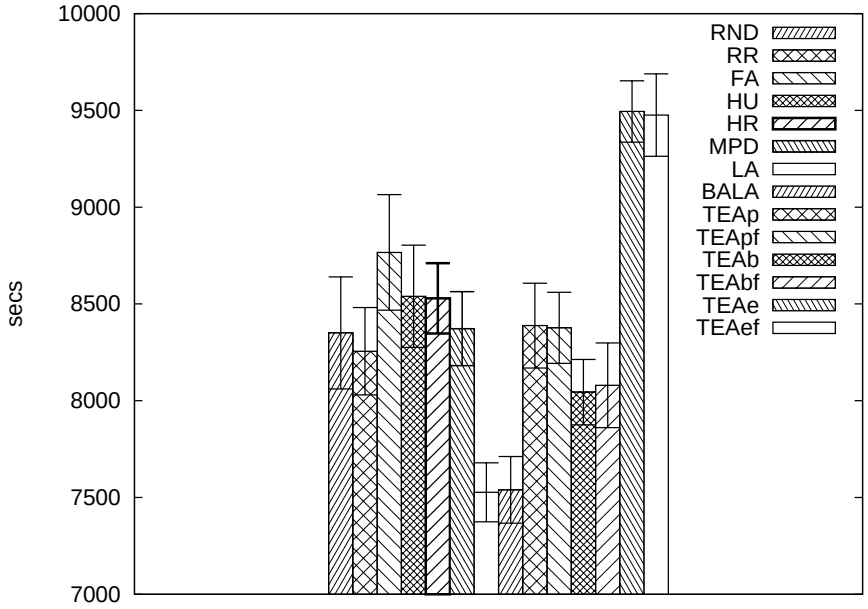

Figura 6.105: Tempo para Conclusão de Cargas: Baixa Prioridade: DC N.Geo. Het., 10 srvs, Binary Tree, Mig, SLA, Comp Usr Def, Mult Pr

as máquinas virtuais, com comportamento de usuários definido e cargas de trabalho com múltiplas prioridades. O algoritmo de escalonamento que apresentou melhores resultados (7.526,70 ± 152,22 s) para este quesito foi o `LA`, empatado tecnicamente pelo menos com o segundo melhor, `BALA` (7.539,38 ± 172,17 s).

A Figura 6.106 traz informações sobre o tempo para conclusão das cargas de trabalho de baixa prioridade desde o recebimento da solicitação pelo *broker*. Neste cenário, avaliamos uma nuvem possuindo 10 servidores distribuídos em um *data center* não geo-

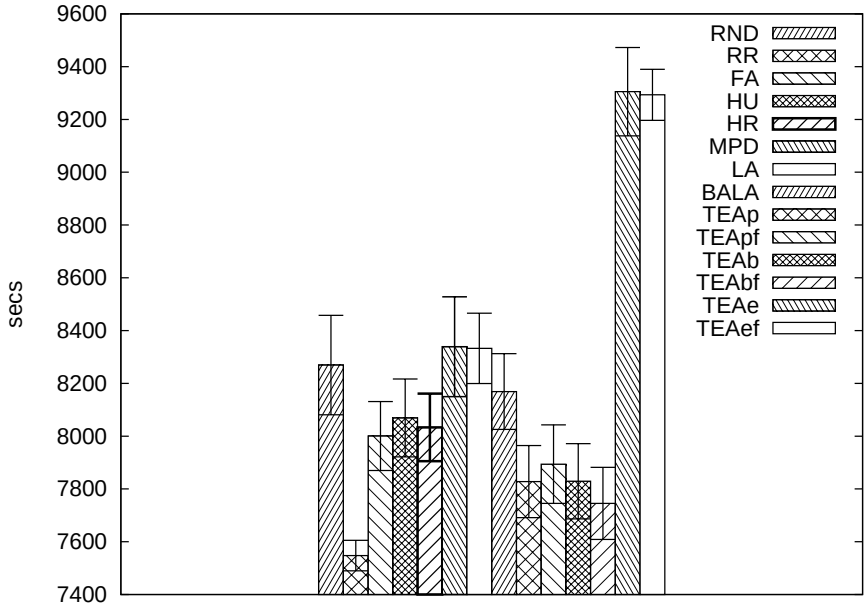

Figura 6.106: Tempo para Conclusão de Cargas: Baixa Prioridade: DC N.Geo. Het., 10 srvs, Binary Tree, S/Mig, SLA, Comp Usr Ind, Mult Pr

distribuído e heterogêneo, empregando topologia *Binary Tree*. O recurso de migração de máquinas virtuais está desativado, com SLA de garantia de 100% de processamento para as máquinas virtuais, com comportamento de usuários indefinido e cargas de trabalho com múltiplas prioridades. O algoritmo de escalonamento que apresentou melhores resultados (7.547,55 ± 57,87 s) para este quesito foi o `RR`, apresentando, considerando o intervalo de confiança, isoladamente os melhores valores do parâmetro comparado dentre todos os algoritmos para este cenário, com uma melhora variando de 0,0% a 5,2% comparado ao segundo melhor (`TEAbf`).

A Figura 6.107 traz informações sobre o tempo para conclusão das cargas de trabalho de baixa prioridade desde o recebimento da solicitação pelo *broker*. Neste cenário, avaliamos uma nuvem possuindo 10 servidores distribuídos em um *data center* não geodistribuído e heterogêneo, empregando topologia *Flat-Tree*. O recurso de migração de máquinas virtuais está desativado, com SLA de garantia de 100% de processamento para as máquinas virtuais, com comportamento de usuários indefinido e cargas de trabalho com múltiplas prioridades. O algoritmo de escalonamento que apresentou melhores resultados (7.416,94 ± 41,26 s) para este quesito foi o `RR`, apresentando, considerando o intervalo de confiança, isoladamente os melhores valores do parâmetro comparado dentre todos os algoritmos para este cenário, com uma melhora variando de 1,6% a 7,1% comparado ao segundo melhor (`TEAbf`).

Apresentamos na Figura 6.108 uma comparação do tempo para conclusão das cargas de trabalho de baixa prioridade desde o recebimento da solicitação pelo *broker*. Neste cenário, avaliamos uma nuvem possuindo 10 servidores distribuídos em um *data center* não geodistribuído e heterogêneo, empregando topologia *K-Ary Fat Tree*. O recurso de migração de máquinas virtuais está desativado, com SLA de garantia de 100% de processamento para as máquinas virtuais, com comportamento de usuários indefinido e cargas

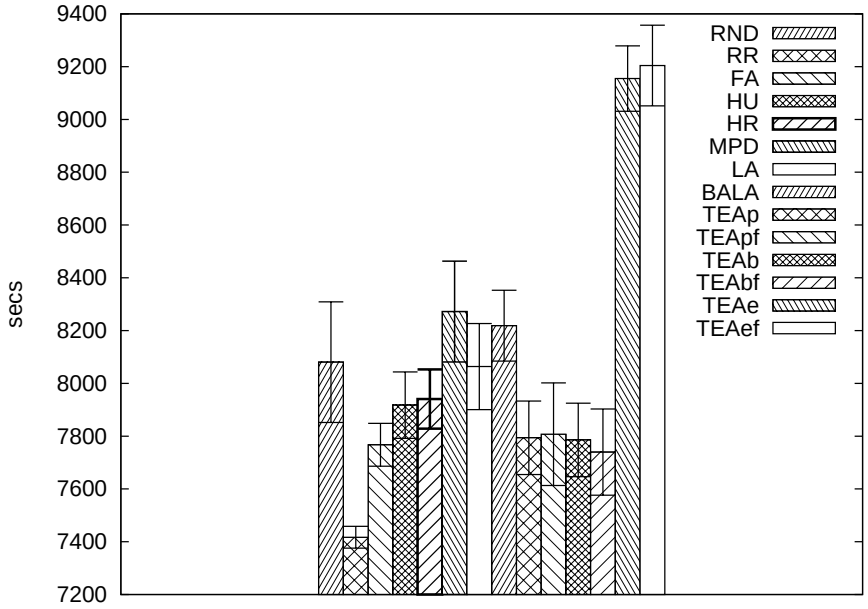

Figura 6.107: Tempo para Conclusão de Cargas: Baixa Prioridade: DC N.Geo. Het., 10 srvs, Flat-Tree, S/Mig, SLA, Comp Usr Ind, Mult Pr

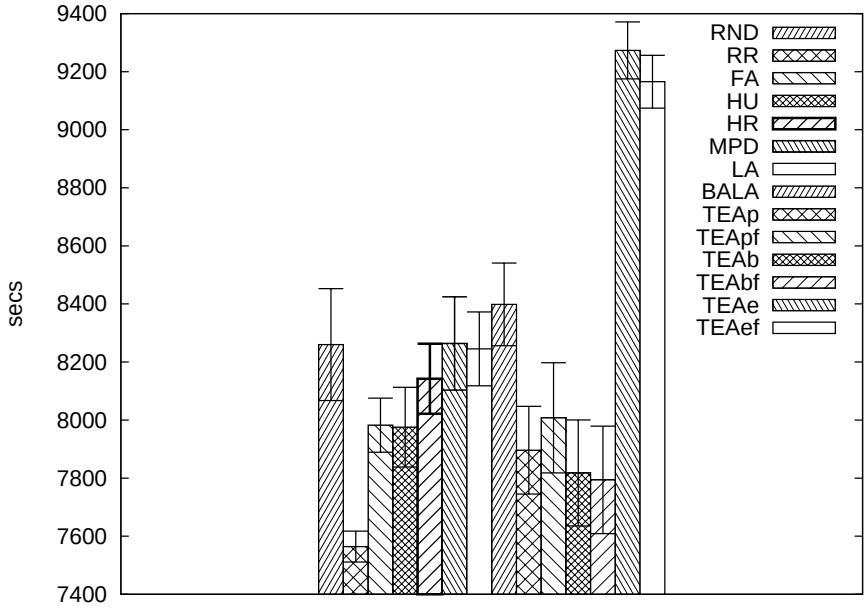

Figura 6.108: Tempo para Conclusão de Cargas: Baixa Prioridade: DC N.Geo. Het., 10 srvs, K-Ary Fat Tree, S/Mig, SLA, Comp Usr Ind, Mult Pr

de trabalho com múltiplas prioridades. O algoritmo de escalonamento que apresentou melhores resultados (7.564,31 ± 53,33 s) para este quesito foi o `RR`, empatado tecnicamente pelo menos com o segundo melhor, `TEAbf` (7.794,12 ± 185,17 s).

A Figura 6.109 traz informações sobre o tempo para conclusão das cargas de trabalho de baixa prioridade desde o recebimento da solicitação pelo *broker*. Neste cenário, avaliamos uma nuvem possuindo 10 servidores distribuídos em um *data center* não geo-distribuído e heterogêneo, empregando topologia *Single-Hop Star*. O recurso de migração

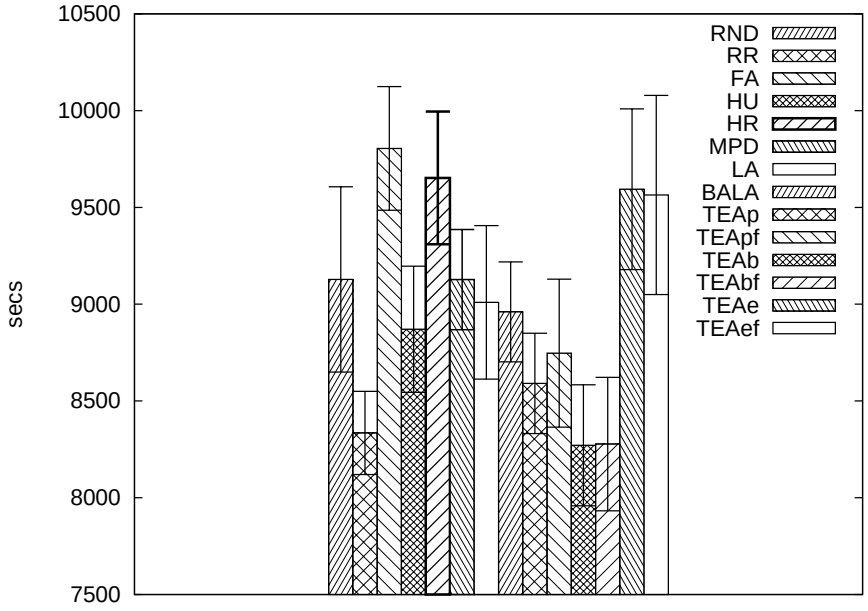

Figura 6.109: Tempo para Conclusão de Cargas: Baixa Prioridade: DC N.Geo. Het., 10 srvs, Single-Hop Star, Mig, S/SLA, Comp Usr Def, Mult Pr

de máquinas virtuais está ativado, sem SLA de garantia de 100% de processamento para as máquinas virtuais, com comportamento de usuários definido e cargas de trabalho com múltiplas prioridades. O algoritmo de escalonamento que apresentou melhores resultados (8.271,12 ± 312,62 s) para este quesito foi o `TEA` (`TEAb`), empatado tecnicamente pelo menos com o segundo melhor, `RR` (8.334,70 ± 215,08 s).

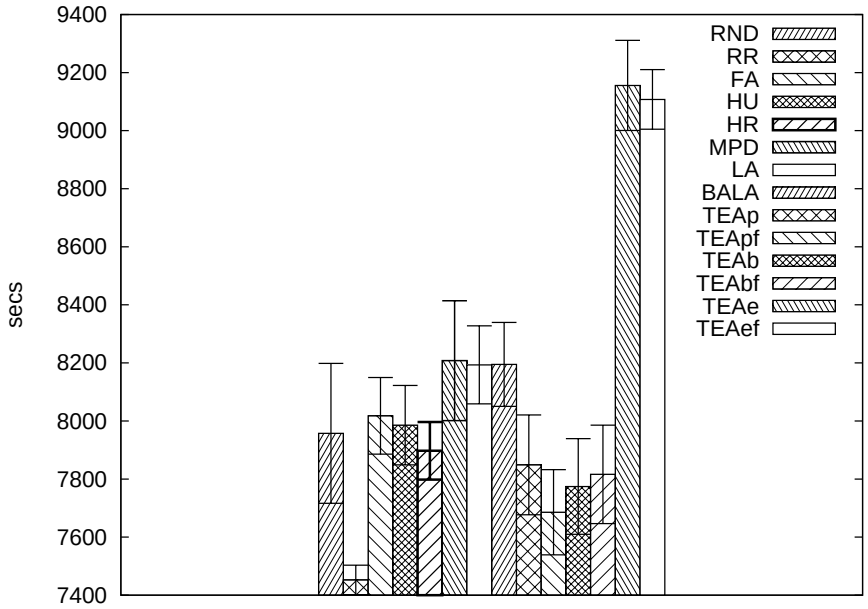

Figura 6.110: Tempo para Conclusão de Cargas: Baixa Prioridade: DC N.Geo. Het., 10 srvs, Single-Hop Star, S/Mig, SLA, Comp Usr Ind, Mult Pr

Apresentamos na Figura 6.110 uma comparação do tempo para conclusão das cargas de trabalho de baixa prioridade desde o recebimento da solicitação pelo *broker*. Neste cenário, avaliamos uma nuvem possuindo 10 servidores distribuídos em um *data center* não geodistribuído e heterogêneo, empregando topologia *Single-Hop Star*. O recurso de migração de máquinas virtuais está desativado, com SLA de garantia de 100% de processamento para as máquinas virtuais, com comportamento de usuários indefinido e cargas de trabalho com múltiplas prioridades. O algoritmo de escalonamento que apresentou melhores resultados (7.452,53 ± 50,99 s) para este quesito foi o `RR`, apresentando, considerando o intervalo de confiança, isoladamente os melhores valores do parâmetro comparado dentre todos os algoritmos para este cenário, com uma melhora variando de 0,5% a 5,8% comparado ao segundo melhor (`TEApf`).

6.4.14 Análise de Resultados: # Migrações Abortadas

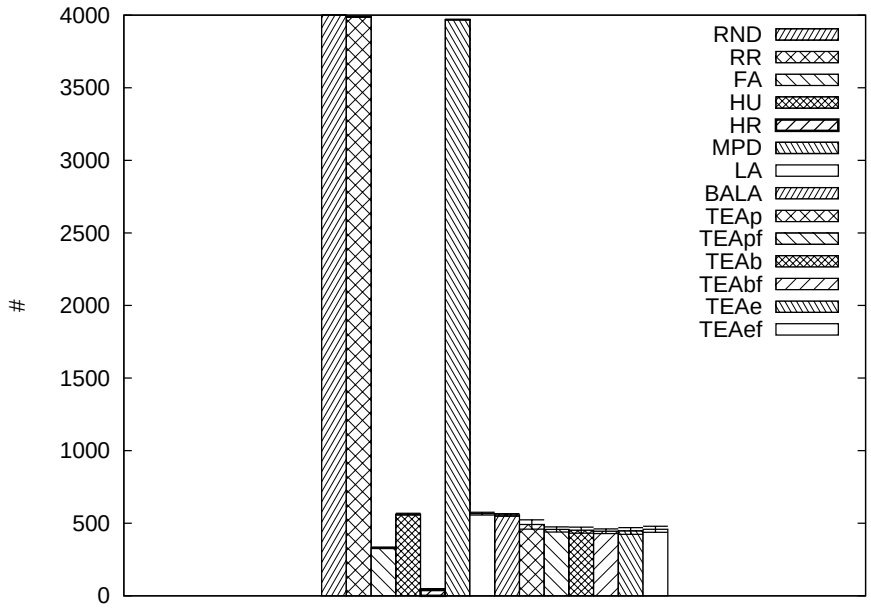

Figura 6.111: # Migrações Abortadas: DC Geo. Homo., 1.000 srvs, Binary Tree, Mig, SLA, Comp Usr Ind, S/Mult Pr

Mostramos na Figura 6.111 um gráfico do número de migrações abortadas. Neste cenário, avaliamos uma nuvem possuindo 1.000 servidores distribuídos em dois *data centers* geodistribuídos e homogêneos, empregando topologia *Binary Tree*. O recurso de migração de máquinas virtuais está ativado, com SLA de garantia de 100% de processamento para as máquinas virtuais, com comportamento de usuários indefinido e cargas de trabalho com somente uma prioridade. O algoritmo de escalonamento que apresentou melhores resultados (42,20 ± 2,52) para este quesito foi o `HR`, apresentando, considerando o intervalo de confiança, isoladamente os melhores valores do parâmetro comparado dentre todos os algoritmos para este cenário, com uma melhora variando de 627,4% a 746,0% comparado ao segundo melhor (`FA`).

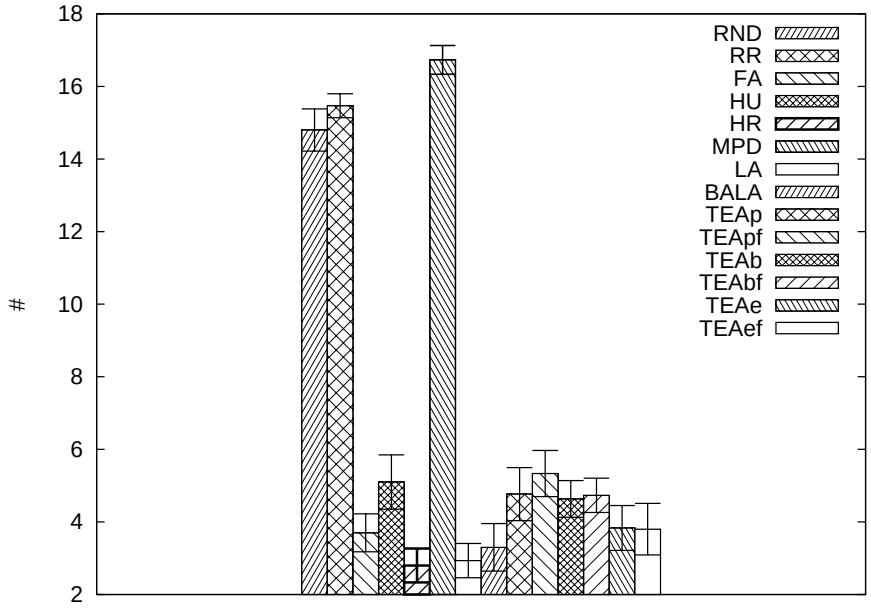

Figura 6.112: # Migrações Abortadas: DC Geo. Het., 10 srvs, Flat-Tree, Mig, SLA, Comp Usr Ind, S/Mult Pr

Mostramos na Figura 6.112 um gráfico do número de migrações abortadas. Neste cenário, avaliamos uma nuvem possuindo 10 servidores distribuídos em dois *data centers* geodistribuídos e heterogêneos, empregando topologia *Flat-Tree*. O recurso de migração de máquinas virtuais está ativado, com SLA de garantia de 100% de processamento para as máquinas virtuais, com comportamento de usuários indefinido e cargas de trabalho com somente uma prioridade. O algoritmo de escalonamento que apresentou melhores resultados (2,80 ± 0,47) para este quesito foi o `HR`, empatado tecnicamente pelo menos com o segundo melhor, `LA` (2,93 ± 0,47).

Mostramos na Figura 6.113 um gráfico do número de migrações abortadas. Neste cenário, avaliamos uma nuvem possuindo 10 servidores distribuídos em dois *data centers* geodistribuídos e heterogêneos, empregando topologia *K-Ary Fat Tree*. O recurso de migração de máquinas virtuais está ativado, com SLA de garantia de 100% de processamento para as máquinas virtuais, com comportamento de usuários indefinido e cargas de trabalho com somente uma prioridade. O algoritmo de escalonamento que apresentou melhores resultados (2,50 ± 0,54) para este quesito foi o `LA`, empatado tecnicamente pelo menos com o segundo melhor, `HR` (2,53 ± 0,45).

A Figura 6.114 traz informações sobre o número de migrações abortadas. Neste cenário, avaliamos uma nuvem possuindo 100 servidores distribuídos em um *data center* não geodistribuído e homogêneo, empregando topologia *Binary Tree*. O recurso de migração de máquinas virtuais está ativado, sem SLA de garantia de 100% de processamento para as máquinas virtuais, com comportamento de usuários definido e cargas de trabalho com múltiplas prioridades. O algoritmo de escalonamento que apresentou melhores resultados (7,73 ± 0,93) para este quesito foi o `TEA` (`TEAbf`), empatado tecnicamente pelo menos com o segundo melhor, `HR` (9,60 ± 1,10).

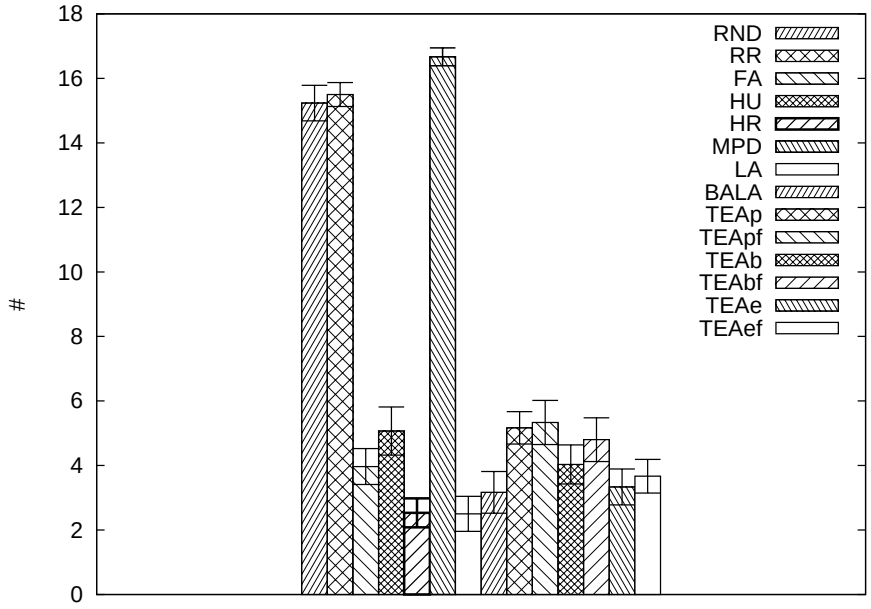

Figura 6.113: # Migrações Abortadas: DC Geo. Het., 10 srvs, K-Ary Fat Tree, Mig, SLA, Comp Usr Ind, S/Mult Pr

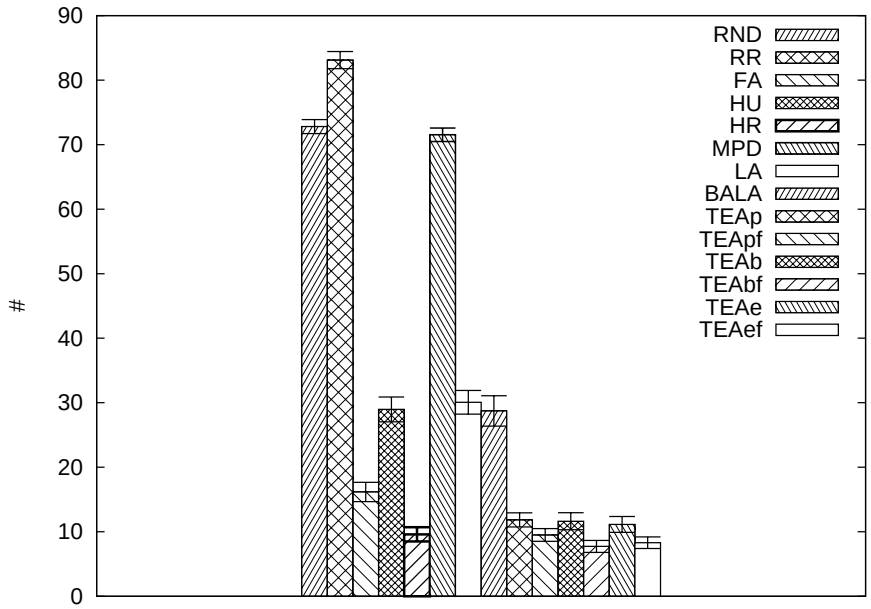

Figura 6.114: # Migrações Abortadas: DC N.Geo. Homo., 100 srvs, Binary Tree, Mig, S/SLA, Comp Usr Def, Mult Pr

A Figura 6.115 traz informações sobre o número de migrações abortadas. Neste cenário, avaliamos uma nuvem possuindo 100 servidores distribuídos em um *data center* não geodistribuído e homogêneo, empregando topologia *Single-Hop Star*. O recurso de migração de máquinas virtuais está ativado, sem SLA de garantia de 100% de processamento para as máquinas virtuais, com comportamento de usuários definido e cargas de trabalho com múltiplas prioridades. O algoritmo de escalonamento que apresentou melhores resul-

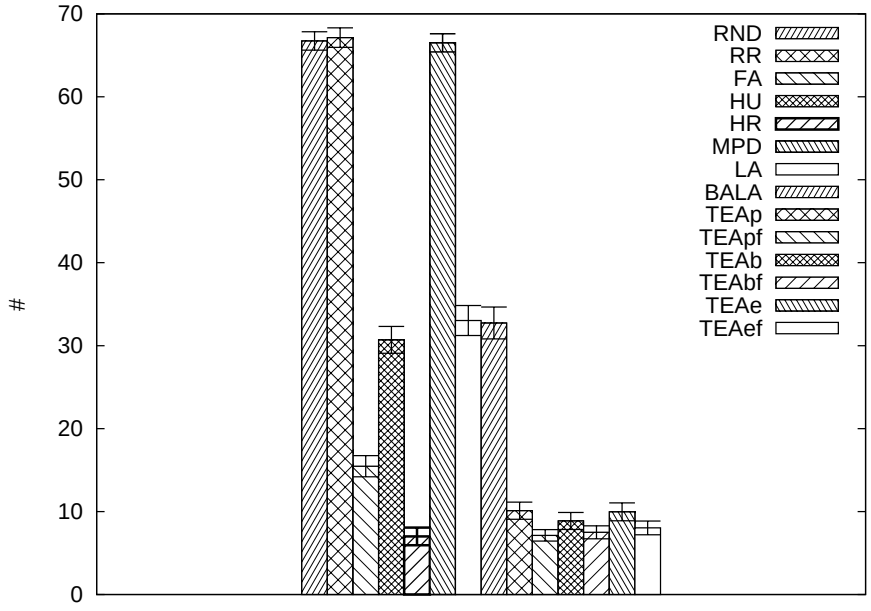

Figura 6.115: # Migrações Abortadas: DC N.Geo. Homo., 100 srvs, Single-Hop Star, Mig, S/SLA, Comp Usr Def, Mult Pr

tados (7,00 ± 1,06) para este quesito foi o `HR`, empatado tecnicamente pelo menos com o segundo melhor, `TEApf` (7,13 ± 0,69).

6.4.15 Análise de Resultados: Discussão

Confrontando os resultados apresentados pelos melhores algoritmos, não observamos violações dos critérios de análise definidos pelas Inequações (3.3), (3.4) e (3.5). Em outras palavras, para todos os algoritmos tidos como os melhores em consumo de energia nos cenários estudados, nenhum apresentou um desempenho de tempo global — *makespan* — inaceitavelmente inferior aos demais. Realizamos na Tabela 6.1 uma análise por características individuais dos *data centers* que compõem a nuvem para os cenários simulados, apresentando valores relativos percentuais do número de vezes em que cada algoritmo apresentou os melhores valores médios para os cenários avaliados, destacando o resultado do melhor algoritmo. Ressaltamos que o total de cenários simulados foi de 10.752, portanto, cada entrada no tabela representa o número de cenários referente ao item avaliado. Por exemplo, temos quatro opções possíveis para topologias, logo cada entrada na tabela indicará o resultado consolidado da simulação de $10.752/4 = 2.688$ cenários.

Tabela 6.1: Análise Excl. por Características da Nuvem

RND	RR	FA	HU	HR	MPD	LA	BALA
TEAp	TEApf	TEAb	TEAbf	TEAe	TEAef	TEA	
			Consumo de Energia Total Médio				
			Data Center Não Geodistribuídos				
0,0%	0,0%	12,2%	0,5%	**21,1%**	7,8%	7,8%	10,4%
2,9%	4,7%	4,2%	4,2%	12,8%	11,5%	40,1%	
			Makespan Médio				
			Data Center Não Geodistribuídos				
0,3%	1,8%	3,1%	0,8%	2,1%	2,1%	4,9%	7,3%
13,0%	12,0%	13,5%	**16,4%**	12,8%	9,9%	77,6%	
			Tempo Médio de Processamento das Cargas				
			Data Center Não Geodistribuídos				
4,7%	**29,4%**	2,3%	2,6%	20,6%	6,0%	9,6%	7,6%
1,8%	3,4%	2,1%	4,2%	2,1%	3,6%	17,2%	
			Tempo Médio de Processamento das Cargas — Alta Prioridade				
			Data Center Não Geodistribuídos				
51,6%	12,2%	1,0%	0,5%	6,8%	0,8%	4,2%	6,0%
2,9%	3,6%	2,9%	2,6%	3,1%	1,8%	16,9%	
			Tempo Médio de Processamento das Cargas — Prioridade Normal				
			Data Center Não Geodistribuídos				
4,4%	**30,2%**	2,3%	3,4%	15,4%	9,1%	6,8%	6,0%
3,4%	4,2%	2,9%	3,6%	3,9%	4,4%	22,4%	
			Tempo Médio de Processamento das Cargas — Baixa Prioridade				
			Data Center Não Geodistribuídos				
53,6%	15,6%	0,0%	1,6%	9,1%	4,7%	3,9%	5,5%
0,3%	1,0%	1,8%	0,8%	1,6%	0,5%	6,0%	
			Tempo Médio para Conclusão das Cargas				
			Data Center Não Geodistribuídos				
0,0%	8,6%	15,1%	1,8%	9,6%	**16,7%**	8,6%	13,0%
4,2%	4,7%	3,9%	4,9%	4,9%	3,9%	26,6%	
			Tempo Médio para Conclusão das Cargas — Alta Prioridade				
			Data Center Não Geodistribuídos				
51,3%	2,9%	6,8%	1,6%	6,8%	5,5%	3,6%	6,0%
1,6%	4,9%	0,8%	2,9%	4,2%	1,3%	15,6%	

RND	RR	FA	HU	HR	MPD	LA	BALA
TEAp	TEApf	TEAb	TEAbf	TEAe	TEAef	TEA	
Tempo Médio para Conclusão das Cargas — Prioridade Normal							
Data Center Não Geodistribuídos							
0,0%	9,6%	14,3%	2,1%	8,1%	**15,9%**	9,9%	13,0%
4,2%	5,2%	3,9%	4,7%	5,7%	3,4%	27,1%	
Tempo Médio para Conclusão das Cargas — Baixa Prioridade							
Data Center Não Geodistribuídos							
50,3%	6,3%	8,3%	1,0%	4,2%	5,2%	5,5%	6,8%
2,3%	1,8%	2,3%	2,9%	1,0%	2,1%	12,5%	
Consumo de Energia Total Médio							
Data Centers Geodistribuídos							
0,0%	0,0%	9,9%	0,3%	8,9%	1,0%	0,0%	0,0%
10,9%	11,7%	8,6%	11,2%	18,2%	**19,3%**	79,9%	
Makespan Médio							
Data Centers Geodistribuídos							
0,0%	0,0%	7,8%	11,2%	3,4%	0,0%	0,0%	0,0%
13,0%	13,3%	8,3%	14,6%	12,8%	**15,6%**	77,6%	
Tempo Médio de Processamento das Cargas							
Data Centers Geodistribuídos							
4,2%	**27,9%**	3,1%	2,6%	22,4%	2,1%	14,3%	13,8%
2,1%	1,8%	1,0%	1,8%	1,3%	1,6%	9,6%	
Tempo Médio de Processamento das Cargas — Alta Prioridade							
Data Centers Geodistribuídos							
54,2%	8,9%	1,6%	1,6%	8,6%	1,8%	6,5%	8,3%
1,6%	0,5%	1,0%	1,6%	2,1%	1,8%	8,6%	
Tempo Médio de Processamento das Cargas — Prioridade Normal							
Data Centers Geodistribuídos							
3,9%	**25,5%**	2,9%	3,1%	21,9%	2,9%	15,4%	14,3%
2,6%	2,6%	1,3%	1,0%	0,8%	1,8%	10,2%	
Tempo Médio de Processamento das Cargas — Baixa Prioridade							
Data Centers Geodistribuídos							
57,0%	11,2%	1,0%	0,8%	9,1%	3,1%	8,3%	3,9%
0,3%	1,6%	1,0%	1,3%	0,8%	0,5%	5,5%	
Tempo Médio para Conclusão das Cargas							
Data Centers Geodistribuídos							
0,0%	0,0%	**19,3%**	14,8%	4,4%	0,3%	0,0%	0,0%
8,3%	11,5%	8,6%	11,7%	9,9%	11,2%	61,2%	

RND	RR	FA	HU	HR	MPD	LA	BALA
TEAp	TEApf	TEAb	TEAbf	TEAe	TEAef	TEA	
Tempo Médio para Conclusão das Cargas — Alta Prioridade							
Data Centers Geodistribuídos							
50,0%	0,0%	4,4%	8,6%	3,1%	3,4%	0,0%	0,0%
5,7%	7,3%	3,9%	4,7%	5,5%	3,4%	30,5%	
Tempo Médio para Conclusão das Cargas — Prioridade Normal							
Data Centers Geodistribuídos							
0,0%	0,0%	**18,0%**	15,6%	4,4%	0,5%	0,0%	0,0%
10,2%	12,0%	8,6%	11,5%	8,6%	10,7%	61,5%	
Tempo Médio para Conclusão das Cargas — Baixa Prioridade							
Data Centers Geodistribuídos							
50,0%	0,0%	9,9%	12,0%	0,8%	0,3%	0,0%	0,0%
4,2%	5,7%	4,2%	3,6%	5,5%	3,9%	27,1%	
Consumo de Energia Total Médio							
Data Centers com topologia Single-Hop Star							
0,0%	0,0%	8,3%	0,5%	14,6%	8,3%	7,8%	5,7%
6,8%	7,8%	5,7%	5,7%	13,5%	**15,1%**	54,7%	
Makespan Médio							
Data Centers com topologia Single-Hop Star							
0,0%	3,6%	3,6%	5,7%	2,6%	4,2%	6,3%	4,7%
9,4%	11,5%	10,9%	**14,6%**	10,9%	12,0%	69,3%	
Tempo Médio de Processamento das Cargas							
Data Centers com topologia Single-Hop Star							
2,6%	**27,6%**	4,2%	3,6%	20,8%	4,7%	14,1%	12,0%
1,0%	3,1%	0,0%	3,1%	1,0%	2,1%	10,4%	
Tempo Médio de Processamento das Cargas — Alta Prioridade							
Data Centers com topologia Single-Hop Star							
52,6%	11,5%	1,0%	2,1%	8,3%	0,5%	5,2%	7,3%
1,0%	2,6%	2,1%	2,1%	2,1%	1,6%	11,5%	
Tempo Médio de Processamento das Cargas — Prioridade Normal							
Data Centers com topologia Single-Hop Star							
3,1%	**27,1%**	4,2%	3,6%	17,7%	6,8%	14,1%	12,5%
2,1%	4,2%	0,0%	2,1%	1,0%	1,6%	10,9%	
Tempo Médio de Processamento das Cargas — Baixa Prioridade							
Data Centers com topologia Single-Hop Star							
53,1%	15,1%	1,0%	0,5%	7,3%	5,2%	5,7%	5,2%
0,0%	2,1%	0,5%	2,1%	2,1%	0,0%	6,8%	

RND	RR	FA	HU	HR	MPD	LA	BALA
TEAp	TEApf	TEAb	TEAbf	TEAe	TEAef	TEA	
Tempo Médio para Conclusão das Cargas							
Data Centers com topologia Single-Hop Star							
0,0%	8,3%	14,1%	7,3%	5,7%	**14,6%**	5,7%	10,4%
3,6%	9,4%	3,1%	6,3%	5,2%	6,3%	33,9%	
Tempo Médio para Conclusão das Cargas — Alta Prioridade							
Data Centers com topologia Single-Hop Star							
50,5%	1,6%	3,1%	4,7%	5,7%	7,3%	2,1%	4,7%
3,1%	4,7%	2,6%	3,1%	4,2%	2,6%	20,3%	
Tempo Médio para Conclusão das Cargas — Prioridade Normal							
Data Centers com topologia Single-Hop Star							
0,0%	9,4%	12,0%	7,3%	4,2%	**14,6%**	5,7%	11,5%
6,3%	9,4%	3,1%	5,7%	5,2%	5,7%	35,4%	
Tempo Médio para Conclusão das Cargas — Baixa Prioridade							
Data Centers com topologia Single-Hop Star							
50,5%	6,3%	6,3%	6,3%	1,0%	6,8%	2,6%	5,2%
2,1%	3,1%	2,1%	2,6%	3,1%	2,1%	15,1%	
Consumo de Energia Total Médio							
Data Centers com topologia Flat-Tree							
0,0%	0,0%	7,3%	0,0%	12,0%	6,3%	5,2%	4,7%
9,9%	7,3%	8,3%	6,8%	**18,2%**	14,1%	64,6%	
Makespan Médio							
Data Centers com topologia Flat-Tree							
0,5%	0,0%	4,7%	7,8%	1,0%	0,0%	0,5%	4,2%
14,1%	11,5%	13,5%	14,6%	**15,1%**	12,5%	81,3%	
Tempo Médio de Processamento das Cargas							
Data Centers com topologia Flat-Tree							
5,7%	**29,2%**	1,6%	2,1%	19,8%	3,1%	17,7%	7,3%
2,6%	2,6%	1,6%	2,6%	2,6%	1,6%	13,5%	
Tempo Médio de Processamento das Cargas — Alta Prioridade							
Data Centers com topologia Flat-Tree							
54,2%	9,9%	2,1%	1,6%	6,8%	1,6%	8,9%	4,2%
2,1%	0,0%	2,1%	1,0%	2,6%	3,1%	10,9%	
Tempo Médio de Processamento das Cargas — Prioridade Normal							
Data Centers com topologia Flat-Tree							
4,7%	**29,2%**	3,1%	2,6%	16,7%	4,7%	14,1%	8,3%
4,2%	3,1%	3,1%	1,6%	2,1%	2,6%	16,7%	

RND	RR	FA	HU	HR	MPD	LA	BALA
TEAp	TEApf	TEAb	TEAbf	TEAe	TEAef	TEA	
Tempo Médio de Processamento das Cargas — Baixa Prioridade							
Data Centers com topologia Flat-Tree							
57,8%	10,4%	0,0%	0,5%	7,8%	3,6%	7,8%	4,7%
0,5%	1,6%	2,6%	1,0%	1,6%	0,0%	7,3%	
Tempo Médio para Conclusão das Cargas							
Data Centers com topologia Flat-Tree							
0,0%	4,2%	**16,1%**	6,3%	6,3%	6,3%	3,1%	6,8%
9,4%	6,3%	6,3%	10,4%	9,9%	8,9%	51,0%	
Tempo Médio para Conclusão das Cargas — Alta Prioridade							
Data Centers com topologia Flat-Tree							
51,0%	2,1%	1,6%	5,2%	4,7%	4,2%	0,5%	3,6%
3,1%	7,8%	2,6%	5,2%	5,7%	2,6%	27,1%	
Tempo Médio para Conclusão das Cargas — Prioridade Normal							
Data Centers com topologia Flat-Tree							
0,0%	5,2%	**14,6%**	7,3%	5,2%	6,3%	3,1%	5,7%
9,4%	7,8%	6,3%	11,5%	10,4%	7,3%	52,6%	
Tempo Médio para Conclusão das Cargas — Baixa Prioridade							
Data Centers com topologia Flat-Tree							
50,0%	2,6%	6,3%	6,8%	3,1%	1,6%	2,6%	3,6%
4,2%	3,1%	3,1%	3,6%	4,2%	5,2%	23,4%	
Consumo de Energia Total Médio							
Data Centers com Topologia Binary Tree							
0,0%	0,0%	13,5%	0,5%	**18,8%**	1,6%	1,6%	6,3%
5,2%	6,8%	5,2%	7,3%	15,6%	17,7%	57,8%	
Makespan Médio							
Data Centers com Topologia Binary Tree							
0,0%	0,0%	7,3%	5,7%	2,6%	0,0%	2,6%	2,6%
12,5%	12,5%	10,4%	**16,7%**	12,0%	15,1%	79,2%	
Tempo Médio de Processamento das Cargas							
Data Centers com Topologia Binary Tree							
5,7%	**29,2%**	3,6%	2,6%	18,8%	4,2%	8,9%	13,0%
2,1%	3,6%	1,6%	3,1%	1,0%	2,6%	14,1%	
Tempo Médio de Processamento das Cargas — Alta Prioridade							
Data Centers com Topologia Binary Tree							
51,6%	11,5%	1,0%	0,0%	6,3%	0,5%	2,6%	10,9%
3,1%	2,1%	2,6%	2,6%	3,6%	1,6%	15,6%	

RND	RR	FA	HU	HR	MPD	LA	BALA
TEAp	TEApf	TEAb	TEAbf	TEAe	TEAef	TEA	
Tempo Médio de Processamento das Cargas — Prioridade Normal							
Data Centers com Topologia Binary Tree							
5,7%	**27,1%**	2,1%	4,7%	16,1%	6,3%	9,9%	10,4%
2,1%	4,7%	1,6%	3,1%	2,6%	3,6%	17,7%	
Tempo Médio de Processamento das Cargas — Baixa Prioridade							
Data Centers com Topologia Binary Tree							
54,2%	15,1%	0,5%	1,6%	8,9%	3,6%	5,2%	5,7%
0,5%	1,0%	1,6%	0,0%	0,5%	1,6%	5,2%	
Tempo Médio para Conclusão das Cargas							
Data Centers com Topologia Binary Tree							
0,0%	2,1%	**21,9%**	9,4%	6,3%	7,3%	3,6%	5,7%
5,7%	6,3%	7,8%	11,5%	4,7%	7,8%	43,8%	
Tempo Médio para Conclusão das Cargas — Alta Prioridade							
Data Centers com Topologia Binary Tree							
50,5%	1,6%	11,5%	3,6%	4,7%	3,1%	1,0%	3,1%
3,6%	4,2%	2,1%	4,2%	4,2%	2,6%	20,8%	
Tempo Médio para Conclusão das Cargas — Prioridade Normal							
Data Centers com Topologia Binary Tree							
0,0%	2,1%	**21,4%**	10,4%	6,8%	6,8%	4,2%	5,7%
7,3%	5,7%	7,8%	9,4%	4,2%	8,3%	42,7%	
Tempo Médio para Conclusão das Cargas — Baixa Prioridade							
Data Centers com Topologia Binary Tree							
50,0%	2,1%	12,0%	6,3%	1,6%	2,1%	2,1%	3,6%
4,2%	3,1%	3,6%	3,1%	3,1%	3,1%	20,3%	
Consumo de Energia Total Médio							
Data Centers com topologia K-Ary Fat Tree							
0,0%	0,0%	**15,1%**	0,5%	14,6%	1,6%	1,0%	4,2%
5,7%	10,9%	6,3%	10,9%	14,6%	14,6%	63,0%	
Makespan Médio							
Data Centers com topologia K-Ary Fat Tree							
0,0%	0,0%	6,3%	4,7%	4,7%	0,0%	0,5%	3,1%
16,1%	15,1%	8,9%	**16,1%**	13,0%	11,5%	80,7%	
Tempo Médio de Processamento das Cargas							
Data Centers com topologia K-Ary Fat Tree							
3,6%	**28,6%**	1,6%	2,1%	26,6%	4,2%	7,3%	10,4%
2,1%	1,0%	3,1%	3,1%	2,1%	4,2%	15,6%	

RND	RR	FA	HU	HR	MPD	LA	BALA
TEAp	TEApf	TEAb	TEAbf	TEAe	TEAef	TEA	
Tempo Médio de Processamento das Cargas — Alta Prioridade							
Data Centers com topologia K-Ary Fat Tree							
53,1%	9,4%	1,0%	0,5%	9,4%	2,6%	4,7%	6,3%
2,6%	3,6%	1,0%	2,6%	2,1%	1,0%	13,0%	
Tempo Médio de Processamento das Cargas — Prioridade Normal							
Data Centers com topologia K-Ary Fat Tree							
3,1%	**28,1%**	1,0%	2,1%	24,0%	6,3%	6,3%	9,4%
3,6%	1,6%	3,6%	2,6%	3,6%	4,7%	19,8%	
Tempo Médio de Processamento das Cargas — Baixa Prioridade							
Data Centers com topologia K-Ary Fat Tree							
56,3%	13,0%	0,5%	2,1%	12,5%	3,1%	5,7%	3,1%
0,0%	0,5%	1,0%	1,0%	0,5%	0,5%	3,6%	
Tempo Médio para Conclusão das Cargas							
Data Centers com topologia K-Ary Fat Tree							
0,0%	2,6%	**16,7%**	10,4%	9,9%	5,7%	4,7%	3,1%
6,3%	10,4%	7,8%	5,2%	9,9%	7,3%	46,9%	
Tempo Médio para Conclusão das Cargas — Alta Prioridade							
Data Centers com topologia K-Ary Fat Tree							
50,5%	0,5%	6,3%	6,8%	4,7%	3,1%	3,6%	0,5%
4,7%	7,8%	2,1%	2,6%	5,2%	1,6%	24,0%	
Tempo Médio para Conclusão das Cargas — Prioridade Normal							
Data Centers com topologia K-Ary Fat Tree							
0,0%	2,6%	**16,7%**	10,4%	8,9%	5,2%	6,8%	3,1%
5,7%	11,5%	7,8%	5,7%	8,9%	6,8%	46,4%	
Tempo Médio para Conclusão das Cargas — Baixa Prioridade							
Data Centers com topologia K-Ary Fat Tree							
50,0%	1,6%	12,0%	6,8%	4,2%	0,5%	3,6%	1,0%
2,6%	5,7%	4,2%	3,6%	2,6%	1,6%	20,3%	
Consumo de Energia Total Médio							
Migração de Máquinas Virtuais Desabilitada							
0,0%	0,0%	4,2%	0,8%	6,3%	8,9%	3,6%	4,7%
7,8%	10,7%	8,3%	8,9%	17,2%	**18,8%**	71,6%	
Makespan Médio							
Migração de Máquinas Virtuais Desabilitada							
0,3%	0,0%	4,2%	3,4%	1,3%	1,8%	1,8%	2,6%
13,0%	14,1%	12,0%	**15,6%**	14,8%	15,1%	84,6%	

RND	RR	FA	HU	HR	MPD	LA	BALA
TEAp	TEApf	TEAb	TEAbf	TEAe	TEAef	TEA	
Tempo Médio de Processamento das Cargas							
Migração de Máquinas Virtuais Desabilitada							
8,1%	**46,4%**	5,5%	3,1%	3,6%	6,3%	3,9%	3,4%
3,1%	3,9%	2,3%	4,4%	2,3%	3,6%	19,8%	
Tempo Médio de Processamento das Cargas — Alta Prioridade							
Migração de Máquinas Virtuais Desabilitada							
55,7%	19,0%	1,0%	1,8%	3,4%	2,6%	2,1%	2,3%
2,1%	2,1%	1,3%	1,8%	2,9%	1,8%	12,0%	
Tempo Médio de Processamento das Cargas — Prioridade Normal							
Migração de Máquinas Virtuais Desabilitada							
6,3%	**44,0%**	4,7%	3,9%	3,9%	7,6%	3,9%	5,2%
4,2%	4,4%	2,6%	2,9%	2,3%	4,2%	20,6%	
Tempo Médio de Processamento das Cargas — Baixa Prioridade							
Migração de Máquinas Virtuais Desabilitada							
56,3%	20,6%	1,0%	1,6%	2,3%	3,4%	1,8%	2,1%
0,5%	2,6%	2,6%	2,1%	2,1%	1,0%	10,9%	
Tempo Médio para Conclusão das Cargas							
Migração de Máquinas Virtuais Desabilitada							
0,0%	2,9%	11,7%	6,3%	8,6%	**13,5%**	4,4%	6,5%
6,0%	8,6%	7,0%	8,1%	7,3%	9,1%	46,1%	
Tempo Médio para Conclusão das Cargas — Alta Prioridade							
Migração de Máquinas Virtuais Desabilitada							
50,3%	1,8%	2,9%	4,7%	2,3%	7,6%	1,8%	2,3%
3,9%	7,0%	2,6%	4,4%	4,9%	3,4%	26,3%	
Tempo Médio para Conclusão das Cargas — Prioridade Normal							
Migração de Máquinas Virtuais Desabilitada							
0,0%	2,3%	10,9%	7,3%	7,8%	**13,5%**	5,2%	6,3%
6,8%	9,4%	7,0%	8,1%	7,0%	8,3%	46,6%	
Tempo Médio para Conclusão das Cargas — Baixa Prioridade							
Migração de Máquinas Virtuais Desabilitada							
50,0%	3,6%	7,0%	5,2%	2,9%	4,4%	2,1%	3,9%
3,1%	4,7%	3,6%	3,1%	3,4%	2,9%	20,8%	
Consumo de Energia Total Médio							
Migração de Máquinas Virtuais Habilitada							
0,0%	0,0%	18,0%	0,0%	**23,7%**	0,0%	4,2%	5,7%
6,0%	5,7%	4,4%	6,5%	13,8%	12,0%	48,4%	

RND	RR	FA	HU	HR	MPD	LA	BALA
TEAp	TEApf	TEAb	TEAbf	TEAe	TEAef	TEA	
			Makespan Médio				
			Migração de Máquinas Virtuais Habilitada				
0,0%	1,8%	6,8%	8,6%	4,2%	0,3%	3,1%	4,7%
13,0%	11,2%	9,9%	**15,4%**	10,7%	10,4%	70,6%	
			Tempo Médio de Processamento das Cargas				
			Migração de Máquinas Virtuais Habilitada				
0,8%	10,9%	0,0%	2,1%	**39,3%**	1,8%	20,1%	18,0%
0,8%	1,3%	0,8%	1,6%	1,0%	1,6%	7,0%	
			Tempo Médio de Processamento das Cargas — Alta Prioridade				
			Migração de Máquinas Virtuais Habilitada				
50,0%	2,1%	1,6%	0,3%	12,0%	0,0%	8,6%	12,0%
2,3%	2,1%	2,6%	2,3%	2,3%	1,8%	13,5%	
			Tempo Médio de Processamento das Cargas — Prioridade Normal				
			Migração de Máquinas Virtuais Habilitada				
2,1%	11,7%	0,5%	2,6%	**33,3%**	4,4%	18,2%	15,1%
1,8%	2,3%	1,6%	1,8%	2,3%	2,1%	12,0%	
			Tempo Médio de Processamento das Cargas — Baixa Prioridade				
			Migração de Máquinas Virtuais Habilitada				
54,4%	6,3%	0,0%	0,8%	15,9%	4,4%	10,4%	7,3%
0,0%	0,0%	0,3%	0,0%	0,3%	0,0%	0,5%	
			Tempo Médio para Conclusão das Cargas				
			Migração de Máquinas Virtuais Habilitada				
0,0%	5,7%	**22,7%**	10,4%	5,5%	3,4%	4,2%	6,5%
6,5%	7,6%	5,5%	8,6%	7,6%	6,0%	41,7%	
			Tempo Médio para Conclusão das Cargas — Alta Prioridade				
			Migração de Máquinas Virtuais Habilitada				
51,0%	1,0%	8,3%	5,5%	7,6%	1,3%	1,8%	3,6%
3,4%	5,2%	2,1%	3,1%	4,7%	1,3%	19,8%	
			Tempo Médio para Conclusão das Cargas — Prioridade Normal				
			Migração de Máquinas Virtuais Habilitada				
0,0%	7,3%	**21,4%**	10,4%	4,7%	2,9%	4,7%	6,8%
7,6%	7,8%	5,5%	8,1%	7,3%	5,7%	41,9%	
			Tempo Médio para Conclusão das Cargas — Baixa Prioridade				
			Migração de Máquinas Virtuais Habilitada				
50,3%	2,6%	11,2%	7,8%	2,1%	1,0%	3,4%	2,9%
3,4%	2,9%	2,9%	3,4%	3,1%	3,1%	18,8%	

RND	RR	FA	HU	HR	MPD	LA	BALA
TEAp	TEApf	TEAb	TEAbf	TEAe	TEAef	TEA	
Consumo de Energia Total Médio							
Data Centers Homogêneos							
0,0%	0,0%	19,5%	0,8%	**23,2%**	5,7%	0,3%	0,0%
7,6%	8,1%	9,4%	8,6%	6,8%	10,2%	50,5%	
Makespan Médio							
Data Centers Homogêneos							
0,3%	1,8%	5,2%	1,0%	1,0%	1,6%	0,3%	0,0%
14,8%	13,8%	10,9%	16,1%	14,6%	**18,5%**	88,8%	
Tempo Médio de Processamento das Cargas							
Data Centers Homogêneos							
4,4%	22,1%	3,1%	3,6%	**24,2%**	4,9%	11,7%	13,8%
0,8%	2,6%	1,6%	2,3%	1,8%	2,9%	12,0%	
Tempo Médio de Processamento das Cargas — Alta Prioridade							
Data Centers Homogêneos							
53,6%	8,3%	2,1%	1,3%	9,9%	0,5%	4,4%	7,6%
2,1%	1,8%	1,6%	2,1%	3,1%	1,6%	12,2%	
Tempo Médio de Processamento das Cargas — Prioridade Normal							
Data Centers Homogêneos							
5,2%	**22,1%**	3,4%	3,9%	21,4%	3,6%	11,5%	13,8%
1,8%	3,6%	2,1%	1,8%	1,8%	3,9%	15,1%	
Tempo Médio de Processamento das Cargas — Baixa Prioridade							
Data Centers Homogêneos							
55,2%	10,7%	0,5%	1,6%	12,2%	3,4%	5,2%	6,0%
0,3%	0,8%	1,0%	1,0%	1,8%	0,3%	5,2%	
Tempo Médio para Conclusão das Cargas							
Data Centers Homogêneos							
0,0%	3,1%	**26,6%**	2,3%	6,3%	12,2%	0,5%	1,0%
7,0%	8,3%	7,0%	8,3%	7,0%	10,2%	47,9%	
Tempo Médio para Conclusão das Cargas — Alta Prioridade							
Data Centers Homogêneos							
50,0%	0,0%	8,1%	1,6%	6,8%	4,9%	0,0%	0,0%
5,2%	7,0%	2,9%	4,9%	6,0%	2,6%	28,6%	
Tempo Médio para Conclusão das Cargas — Prioridade Normal							
Data Centers Homogêneos							
0,0%	4,7%	**24,5%**	3,9%	3,9%	12,2%	1,3%	0,8%
7,8%	9,1%	7,6%	8,3%	6,8%	9,1%	48,7%	

RND	RR	FA	HU	HR	MPD	LA	BALA
TEAp	TEApf	TEAb	TEAbf	TEAe	TEAef	TEA	
Tempo Médio para Conclusão das Cargas — Baixa Prioridade							
Data Centers Homogêneos							
50,3%	3,1%	13,5%	1,0%	3,9%	4,7%	0,3%	0,8%
3,6%	3,6%	3,6%	3,4%	4,4%	3,6%	22,4%	
Consumo de Energia Total Médio							
Data Centers Heterogêneos							
0,0%	0,0%	2,6%	0,0%	6,8%	3,1%	7,6%	10,4%
6,3%	8,3%	3,4%	6,8%	**24,2%**	20,6%	69,5%	
Makespan Médio							
Data Centers Heterogêneos							
0,0%	0,0%	5,7%	10,9%	4,4%	0,5%	4,7%	7,3%
11,2%	11,5%	10,9%	**14,8%**	10,9%	7,0%	66,4%	
Tempo Médio de Processamento das Cargas							
Data Centers Heterogêneos							
4,4%	**35,2%**	2,3%	1,6%	18,8%	3,1%	12,2%	7,6%
3,1%	2,6%	1,6%	3,6%	1,6%	2,3%	14,8%	
Tempo Médio de Processamento das Cargas — Alta Prioridade							
Data Centers Heterogêneos							
52,1%	12,8%	0,5%	0,8%	5,5%	2,1%	6,3%	6,8%
2,3%	2,3%	2,3%	2,1%	2,1%	2,1%	13,3%	
Tempo Médio de Processamento das Cargas — Prioridade Normal							
Data Centers Heterogêneos							
3,1%	**33,6%**	1,8%	2,6%	15,9%	8,3%	10,7%	6,5%
4,2%	3,1%	2,1%	2,9%	2,9%	2,3%	17,4%	
Tempo Médio de Processamento das Cargas — Baixa Prioridade							
Data Centers Heterogêneos							
55,5%	16,1%	0,5%	0,8%	6,0%	4,4%	7,0%	3,4%
0,3%	1,8%	1,8%	1,0%	0,5%	0,8%	6,3%	
Tempo Médio para Conclusão das Cargas							
Data Centers Heterogêneos							
0,0%	5,5%	7,8%	**14,3%**	7,8%	4,7%	8,1%	12,0%
5,5%	7,8%	5,5%	8,3%	7,8%	4,9%	39,8%	
Tempo Médio para Conclusão das Cargas — Alta Prioridade							
Data Centers Heterogêneos							
51,3%	2,9%	3,1%	8,6%	3,1%	3,9%	3,6%	6,0%
2,1%	5,2%	1,8%	2,6%	3,6%	2,1%	17,4%	

RND	RR	FA	HU	HR	MPD	LA	BALA
TEAp	TEApf	TEAb	TEAbf	TEAe	TEAef	TEA	
Tempo Médio para Conclusão das Cargas — Prioridade Normal							
Data Centers Heterogêneos							
0,0%	4,9%	7,8%	**13,8%**	8,6%	4,2%	8,6%	12,2%
6,5%	8,1%	4,9%	7,8%	7,6%	4,9%	39,8%	
Tempo Médio para Conclusão das Cargas — Baixa Prioridade							
Data Centers Heterogêneos							
50,0%	3,1%	4,7%	12,0%	1,0%	0,8%	5,2%	6,0%
2,9%	3,9%	2,9%	3,1%	2,1%	2,3%	17,2%	
Consumo de Energia Total Médio							
Nuvem com 10 servidores							
0,0%	0,0%	25,8%	0,4%	3,5%	0,0%	0,0%	0,0%
3,9%	3,1%	3,1%	4,7%	27,3%	**28,1%**	70,3%	
Makespan Médio							
Nuvem com 10 servidores							
0,4%	0,4%	12,5%	12,1%	5,5%	0,0%	1,6%	2,0%
10,2%	7,8%	12,9%	**16,8%**	8,6%	9,4%	65,6%	
Tempo Médio de Processamento das Cargas							
Nuvem com 10 servidores							
5,1%	**25,4%**	3,1%	5,1%	20,7%	9,4%	8,6%	10,9%
1,2%	3,1%	1,2%	2,3%	1,6%	2,3%	11,7%	
Tempo Médio de Processamento das Cargas — Alta Prioridade							
Nuvem com 10 servidores							
53,1%	4,3%	3,1%	2,0%	2,7%	1,2%	5,5%	8,6%
2,7%	3,9%	2,7%	3,5%	3,9%	2,7%	19,5%	
Tempo Médio de Processamento das Cargas — Prioridade Normal							
Nuvem com 10 servidores							
2,7%	**22,7%**	3,5%	6,6%	13,3%	11,7%	9,8%	13,7%
2,3%	4,7%	1,2%	2,0%	2,3%	3,5%	16,0%	
Tempo Médio de Processamento das Cargas — Baixa Prioridade							
Nuvem com 10 servidores							
59,4%	13,3%	0,8%	2,3%	3,9%	8,6%	3,5%	2,3%
0,8%	0,8%	0,8%	2,3%	0,8%	0,4%	5,9%	
Tempo Médio para Conclusão das Cargas							
Nuvem com 10 servidores							
0,0%	9,8%	**37,1%**	16,4%	17,2%	9,0%	3,1%	3,1%
0,8%	0,8%	1,2%	1,6%	0,0%	0,0%	4,3%	

RND	RR	FA	HU	HR	MPD	LA	BALA
TEAp	TEApf	TEAb	TEAbf	TEAe	TEAef	TEA	
Tempo Médio para Conclusão das Cargas — Alta Prioridade							
Nuvem com 10 servidores							
52,0%	4,3%	7,8%	11,3%	8,6%	0,8%	0,0%	0,4%
2,3%	5,5%	1,2%	2,0%	3,5%	0,4%	14,8%	
Tempo Médio para Conclusão das Cargas — Prioridade Normal							
Nuvem com 10 servidores							
0,0%	11,3%	**34,0%**	18,8%	14,5%	9,4%	3,9%	3,1%
2,3%	1,2%	0,4%	1,2%	0,0%	0,0%	5,1%	
Tempo Médio para Conclusão das Cargas — Baixa Prioridade							
Nuvem com 10 servidores							
50,4%	7,0%	18,0%	10,2%	3,9%	1,2%	2,3%	2,3%
0,8%	0,8%	1,2%	2,0%	0,0%	0,0%	4,7%	
Consumo de Energia Total Médio							
Nuvem com 100 servidores							
0,0%	0,0%	2,0%	0,8%	**28,1%**	9,0%	9,0%	13,7%
5,1%	6,3%	6,6%	6,6%	6,3%	6,6%	37,5%	
Makespan Médio							
Nuvem com 100 servidores							
0,0%	0,0%	0,8%	5,9%	0,0%	0,0%	3,5%	6,6%
11,3%	14,8%	10,2%	15,6%	**16,0%**	15,2%	83,2%	
Tempo Médio de Processamento das Cargas							
Nuvem com 100 servidores							
1,2%	**32,4%**	3,1%	1,2%	19,9%	1,6%	11,3%	13,3%
2,3%	2,0%	2,7%	3,5%	1,6%	3,9%	16,0%	
Tempo Médio de Processamento das Cargas — Alta Prioridade							
Nuvem com 100 servidores							
52,7%	14,8%	0,0%	0,8%	9,0%	1,2%	3,5%	6,6%
2,0%	1,6%	2,0%	2,0%	2,0%	2,0%	11,3%	
Tempo Médio de Processamento das Cargas — Prioridade Normal							
Nuvem com 100 servidores							
5,1%	**33,2%**	2,0%	1,6%	18,8%	4,7%	8,2%	10,2%
2,3%	2,3%	3,5%	2,0%	2,7%	3,5%	16,4%	
Tempo Médio de Processamento das Cargas — Baixa Prioridade							
Nuvem com 100 servidores							
52,7%	14,8%	0,4%	0,4%	9,4%	2,3%	7,4%	7,8%
0,0%	1,6%	0,4%	0,4%	2,0%	0,4%	4,7%	

RND	RR	FA	HU	HR	MPD	LA	BALA
TEAp	TEApf	TEAb	TEAbf	TEAe	TEAef	TEA	
Tempo Médio para Conclusão das Cargas							
Nuvem com 100 servidores							
0,0%	0,0%	9,0%	8,6%	2,3%	**12,5%**	8,2%	**12,5%**
7,0%	8,6%	6,6%	10,2%	7,0%	7,4%	46,9%	
Tempo Médio para Conclusão das Cargas — Alta Prioridade							
Nuvem com 100 servidores							
50,0%	0,0%	3,1%	2,3%	2,7%	10,2%	4,7%	7,8%
2,7%	2,7%	2,3%	4,3%	5,1%	2,0%	19,1%	
Tempo Médio para Conclusão das Cargas — Prioridade Normal							
Nuvem com 100 servidores							
0,0%	0,0%	8,2%	7,8%	2,0%	11,3%	9,0%	**12,9%**
8,6%	8,6%	7,0%	8,6%	8,2%	7,8%	48,8%	
Tempo Médio para Conclusão das Cargas — Baixa Prioridade							
Nuvem com 100 servidores							
50,0%	0,8%	5,5%	9,4%	2,0%	5,5%	4,7%	5,9%
1,6%	2,7%	3,9%	3,5%	2,7%	2,0%	16,4%	
Consumo de Energia Total Médio							
Nuvem com 1.000 servidores							
0,0%	0,0%	5,5%	0,0%	13,3%	4,3%	2,7%	2,0%
11,7%	**15,2%**	9,4%	11,7%	12,9%	11,3%	72,3%	
Makespan Médio							
Nuvem com 1.000 servidores							
0,0%	2,3%	3,1%	0,0%	2,7%	3,1%	2,3%	2,3%
17,6%	15,2%	9,8%	14,1%	13,7%	13,7%	84,0%	
Tempo Médio de Processamento das Cargas							
Nuvem com 1.000 servidores							
7,0%	**28,1%**	2,0%	1,6%	23,8%	1,2%	16,0%	7,8%
2,3%	2,7%	0,8%	3,1%	2,0%	1,6%	12,5%	
Tempo Médio de Processamento das Cargas — Alta Prioridade							
Nuvem com 1.000 servidores							
52,7%	12,5%	0,8%	0,4%	11,3%	1,6%	7,0%	6,3%
2,0%	0,8%	1,2%	0,8%	2,0%	0,8%	7,4%	
Tempo Médio de Processamento das Cargas — Prioridade Normal							
Nuvem com 1.000 servidores							
4,7%	**27,7%**	2,3%	1,6%	23,8%	1,6%	15,2%	6,6%
4,3%	3,1%	1,6%	3,1%	2,0%	2,3%	16,4%	

RND	RR	FA	HU	HR	MPD	LA	BALA
TEAp	TEApf	TEAb	TEAbf	TEAe	TEAef	TEA	
Tempo Médio de Processamento das Cargas — Baixa Prioridade							
Nuvem com 1.000 servidores							
53,9%	12,1%	0,4%	0,8%	14,1%	0,8%	7,4%	3,9%
0,0%	1,6%	3,1%	0,4%	0,8%	0,8%	6,6%	
Tempo Médio para Conclusão das Cargas							
Nuvem com 1.000 servidores							
0,0%	3,1%	5,5%	0,0%	1,6%	3,9%	1,6%	3,9%
10,9%	14,8%	10,9%	13,3%	**15,2%**	**15,2%**	80,5%	
Tempo Médio para Conclusão das Cargas — Alta Prioridade							
Nuvem com 1.000 servidores							
50,0%	0,0%	5,9%	1,6%	3,5%	2,3%	0,8%	0,8%
5,9%	10,2%	3,5%	5,1%	5,9%	4,7%	35,2%	
Tempo Médio para Conclusão das Cargas — Prioridade Normal							
Nuvem com 1.000 servidores							
0,0%	3,1%	6,3%	0,0%	2,3%	3,9%	2,0%	3,5%
10,5%	**16,0%**	11,3%	14,5%	13,3%	13,3%	78,9%	
Tempo Médio para Conclusão das Cargas — Baixa Prioridade							
Nuvem com 1.000 servidores							
50,0%	1,6%	3,9%	0,0%	1,6%	1,6%	1,2%	2,0%
7,4%	7,8%	4,7%	4,3%	7,0%	7,0%	38,3%	
Consumo de Energia Total Médio							
Nuvem sem SLA de Garantia de 100% de Processamento para as Máquinas Virtuais							
0,0%	0,0%	13,0%	0,3%	9,6%	4,2%	4,4%	4,9%
7,6%	8,3%	6,8%	8,1%	14,6%	**18,2%**	63,5%	
Makespan Médio							
Nuvem sem SLA de Garantia de 100% de Processamento para as Máquinas Virtuais							
0,3%	0,8%	4,4%	8,1%	2,3%	0,8%	3,4%	5,5%
14,3%	12,8%	8,9%	**15,4%**	12,5%	10,7%	74,5%	
Tempo Médio de Processamento das Cargas							
Nuvem sem SLA de Garantia de 100% de Processamento para as Máquinas Virtuais							
4,2%	**53,4%**	0,0%	0,0%	6,3%	5,5%	15,1%	15,6%
0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	
Tempo Médio de Processamento das Cargas — Alta Prioridade							
Nuvem sem SLA de Garantia de 100% de Processamento para as Máquinas Virtuais							
53,9%	18,5%	1,0%	0,3%	3,4%	0,5%	7,3%	12,2%
0,5%	1,0%	0,3%	0,5%	0,5%	0,0%	2,9%	

RND	RR	FA	HU	HR	MPD	LA	BALA
TEAp	TEApf	TEAb	TEAbf	TEAe	TEAef	TEA	
Tempo Médio de Processamento das Cargas — Prioridade Normal							
Nuvem sem SLA de Garantia de 100% de Processamento para as Máquinas Virtuais							
4,9%	**51,8%**	0,0%	0,0%	6,3%	7,6%	14,1%	15,4%
0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	
Tempo Médio de Processamento das Cargas — Baixa Prioridade							
Nuvem sem SLA de Garantia de 100% de Processamento para as Máquinas Virtuais							
56,3%	22,7%	0,0%	0,3%	3,6%	3,6%	8,1%	5,5%
0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	
Tempo Médio para Conclusão das Cargas							
Nuvem sem SLA de Garantia de 100% de Processamento para as Máquinas Virtuais							
0,0%	4,9%	**16,1%**	9,6%	3,9%	8,9%	4,9%	6,5%
5,7%	7,0%	6,3%	9,1%	7,3%	9,6%	45,1%	
Tempo Médio para Conclusão das Cargas — Alta Prioridade							
Nuvem sem SLA de Garantia de 100% de Processamento para as Máquinas Virtuais							
50,0%	1,0%	6,3%	4,2%	2,6%	5,7%	1,8%	2,9%
3,6%	7,0%	3,1%	4,2%	4,7%	2,9%	25,5%	
Tempo Médio para Conclusão das Cargas — Prioridade Normal							
Nuvem sem SLA de Garantia de 100% de Processamento para as Máquinas Virtuais							
0,0%	4,9%	**14,8%**	10,7%	3,4%	8,9%	6,0%	6,3%
7,6%	7,6%	5,7%	8,9%	6,8%	8,6%	45,1%	
Tempo Médio para Conclusão das Cargas — Baixa Prioridade							
Nuvem sem SLA de Garantia de 100% de Processamento para as Máquinas Virtuais							
50,0%	3,1%	7,8%	6,8%	1,8%	3,1%	2,1%	3,4%
3,9%	3,6%	3,9%	3,6%	2,9%	3,9%	21,9%	
Consumo de Energia Total Médio							
Nuvem com SLA de Garantia de 100% de Processamento para as Máquinas Virtuais							
0,0%	0,0%	9,1%	0,5%	**20,3%**	4,7%	3,4%	5,5%
6,3%	8,1%	6,0%	7,3%	16,4%	12,5%	56,5%	
Makespan Médio							
Nuvem com SLA de Garantia de 100% de Processamento para as Máquinas Virtuais							
0,0%	1,0%	6,5%	3,9%	3,1%	1,3%	1,6%	1,8%
11,7%	12,5%	13,0%	**15,6%**	13,0%	14,8%	80,7%	
Tempo Médio de Processamento das Cargas							
Nuvem com SLA de Garantia de 100% de Processamento para as Máquinas Virtuais							
4,7%	3,9%	5,5%	5,2%	**36,7%**	2,6%	8,9%	5,7%
3,9%	5,2%	3,1%	6,0%	3,4%	5,2%	26,8%	

RND	RR	FA	HU	HR	MPD	LA	BALA
TEAp	TEApf	TEAb	TEAbf	TEAe	TEAef	TEA	
Tempo Médio de Processamento das Cargas — Alta Prioridade							
Nuvem com SLA de Garantia de 100% de Processamento para as Máquinas Virtuais							
51,8%	2,6%	1,6%	1,8%	12,0%	2,1%	3,4%	2,1%
3,9%	3,1%	3,6%	3,6%	4,7%	3,6%	22,7%	
Tempo Médio de Processamento das Cargas — Prioridade Normal							
Nuvem com SLA de Garantia de 100% de Processamento para as Máquinas Virtuais							
3,4%	3,9%	5,2%	6,5%	**31,0%**	4,4%	8,1%	4,9%
6,0%	6,8%	4,2%	4,7%	4,7%	6,3%	32,6%	
Tempo Médio de Processamento das Cargas — Baixa Prioridade							
Nuvem com SLA de Garantia de 100% de Processamento para as Máquinas Virtuais							
54,4%	4,2%	1,0%	2,1%	14,6%	4,2%	4,2%	3,9%
0,5%	2,6%	2,9%	2,1%	2,3%	1,0%	11,5%	
Tempo Médio para Conclusão das Cargas							
Nuvem com SLA de Garantia de 100% de Processamento para as Máquinas Virtuais							
0,0%	3,6%	**18,2%**	7,0%	10,2%	8,1%	3,6%	6,5%
6,8%	9,1%	6,3%	7,6%	7,6%	5,5%	42,7%	
Tempo Médio para Conclusão das Cargas — Alta Prioridade							
Nuvem com SLA de Garantia de 100% de Processamento para as Máquinas Virtuais							
51,3%	1,8%	4,9%	6,0%	7,3%	3,1%	1,8%	3,1%
3,6%	5,2%	1,6%	3,4%	4,9%	1,8%	20,6%	
Tempo Médio para Conclusão das Cargas — Prioridade Normal							
Nuvem com SLA de Garantia de 100% de Processamento para as Máquinas Virtuais							
0,0%	4,7%	**17,4%**	7,0%	9,1%	7,6%	3,9%	6,8%
6,8%	9,6%	6,8%	7,3%	7,6%	5,5%	43,5%	
Tempo Médio para Conclusão das Cargas — Baixa Prioridade							
Nuvem com SLA de Garantia de 100% de Processamento para as Máquinas Virtuais							
50,3%	3,1%	10,4%	6,3%	3,1%	2,3%	3,4%	3,4%
2,6%	3,9%	2,6%	2,9%	3,6%	2,1%	17,7%	
Consumo de Energia Total Médio							
Comportamento de Usuários Indefinido							
0,0%	0,0%	11,5%	0,3%	15,1%	4,2%	4,4%	4,7%
8,9%	6,8%	5,7%	7,6%	14,8%	**16,1%**	59,9%	
Makespan Médio							
Comportamento de Usuários Indefinido							
0,0%	1,3%	5,7%	6,5%	1,8%	1,6%	2,1%	4,2%
15,9%	10,9%	10,4%	15,6%	12,8%	11,2%	76,8%	

RND	RR	FA	HU	HR	MPD	LA	BALA
TEAp	TEApf	TEAb	TEAbf	TEAe	TEAef	TEA	
			Tempo Médio de Processamento das Cargas				
			Comportamento de Usuários Indefinido				
4,7%	**29,4%**	2,9%	2,1%	21,9%	3,6%	10,7%	12,0%
2,3%	2,1%	1,3%	2,6%	2,3%	2,1%	12,8%	
			Tempo Médio de Processamento das Cargas — Alta Prioridade				
			Comportamento de Usuários Indefinido				
53,1%	10,9%	0,8%	1,0%	8,6%	0,8%	5,2%	7,3%
2,3%	1,8%	1,3%	1,3%	3,9%	1,6%	12,2%	
			Tempo Médio de Processamento das Cargas — Prioridade Normal				
			Comportamento de Usuários Indefinido				
3,6%	**27,3%**	2,6%	3,1%	18,8%	6,3%	9,1%	11,7%
4,2%	4,2%	1,8%	2,3%	2,9%	2,1%	17,4%	
			Tempo Médio de Processamento das Cargas — Baixa Prioridade				
			Comportamento de Usuários Indefinido				
54,4%	14,3%	0,8%	1,8%	9,1%	4,2%	4,9%	5,2%
0,0%	1,6%	0,5%	1,6%	0,8%	0,8%	5,2%	
			Tempo Médio para Conclusão das Cargas				
			Comportamento de Usuários Indefinido				
0,0%	3,4%	**17,4%**	9,1%	7,0%	8,1%	3,9%	7,0%
6,8%	7,3%	7,3%	8,9%	7,8%	6,0%	44,0%	
			Tempo Médio para Conclusão das Cargas — Alta Prioridade				
			Comportamento de Usuários Indefinido				
50,8%	1,0%	5,7%	5,5%	4,2%	4,7%	1,8%	2,9%
2,9%	5,5%	3,6%	3,6%	4,9%	2,9%	23,4%	
			Tempo Médio para Conclusão das Cargas — Prioridade Normal				
			Comportamento de Usuários Indefinido				
0,0%	4,4%	**15,6%**	9,6%	6,0%	7,8%	4,9%	7,0%
7,6%	7,8%	7,8%	8,3%	7,6%	5,5%	44,5%	
			Tempo Médio para Conclusão das Cargas — Baixa Prioridade				
			Comportamento de Usuários Indefinido				
50,3%	2,6%	9,6%	6,8%	2,3%	2,1%	2,1%	3,9%
4,2%	3,1%	3,4%	3,6%	3,4%	2,6%	20,3%	
			Consumo de Energia Total Médio				
			Comportamento de Usuários Definido				
0,0%	0,0%	10,7%	0,5%	14,8%	4,7%	3,4%	5,7%
4,9%	9,6%	7,0%	7,8%	**16,1%**	14,6%	60,2%	

RND	RR	FA	HU	HR	MPD	LA	BALA
TEAp	TEApf	TEAb	TEAbf	TEAe	TEAef	TEA	
Makespan Médio							
Comportamento de Usuários Definido							
0,3%	0,5%	5,2%	5,5%	3,6%	0,5%	2,9%	3,1%
10,2%	14,3%	11,5%	**15,4%**	12,8%	14,3%	78,4%	
Tempo Médio de Processamento das Cargas							
Comportamento de Usuários Definido							
4,2%	**27,9%**	2,6%	3,1%	21,1%	4,4%	13,3%	9,4%
1,6%	3,1%	1,8%	3,4%	1,0%	3,1%	14,1%	
Tempo Médio de Processamento das Cargas — Alta Prioridade							
Comportamento de Usuários Definido							
52,6%	10,2%	1,8%	1,0%	6,8%	1,8%	5,5%	7,0%
2,1%	2,3%	2,6%	2,9%	1,3%	2,1%	13,3%	
Tempo Médio de Processamento das Cargas — Prioridade Normal							
Comportamento de Usuários Definido							
4,7%	**28,4%**	2,6%	3,4%	18,5%	5,7%	13,0%	8,6%
1,8%	2,6%	2,3%	2,3%	1,8%	4,2%	15,1%	
Tempo Médio de Processamento das Cargas — Baixa Prioridade							
Comportamento de Usuários Definido							
56,3%	12,5%	0,3%	0,5%	9,1%	3,6%	7,3%	4,2%
0,5%	1,0%	2,3%	0,5%	1,6%	0,3%	6,3%	
Tempo Médio para Conclusão das Cargas							
Comportamento de Usuários Definido							
0,0%	5,2%	**16,9%**	7,6%	7,0%	8,9%	4,7%	6,0%
5,7%	8,9%	5,2%	7,8%	7,0%	9,1%	43,8%	
Tempo Médio para Conclusão das Cargas — Alta Prioridade							
Comportamento de Usuários Definido							
50,5%	1,8%	5,5%	4,7%	5,7%	4,2%	1,8%	3,1%
4,4%	6,8%	1,0%	3,9%	4,7%	1,8%	22,7%	
Tempo Médio para Conclusão das Cargas — Prioridade Normal							
Comportamento de Usuários Definido							
0,0%	5,2%	**16,7%**	8,1%	6,5%	8,6%	4,9%	6,0%
6,8%	9,4%	4,7%	7,8%	6,8%	8,6%	44,0%	
Tempo Médio para Conclusão das Cargas — Baixa Prioridade							
Comportamento de Usuários Definido							
50,0%	3,6%	8,6%	6,3%	2,6%	3,4%	3,4%	2,9%
2,3%	4,4%	3,1%	2,9%	3,1%	3,4%	19,3%	

RND	RR	FA	HU	HR	MPD	LA	BALA
TEAp	TEApf	TEAb	TEAbf	TEAe	TEAef	TEA	
Consumo de Energia Total Médio							
Máquinas Virtuais Possuindo Somente uma Prioridade							
0,0%	0,0%	8,3%	0,3%	13,0%	3,4%	4,4%	4,7%
8,9%	9,9%	7,0%	8,1%	**17,2%**	14,8%	65,9%	
Makespan Médio							
Máquinas Virtuais Possuindo Somente uma Prioridade							
0,0%	1,0%	1,8%	2,1%	1,8%	1,6%	1,8%	3,1%
14,3%	14,8%	12,2%	**17,4%**	14,3%	13,5%	86,7%	
Tempo Médio de Processamento das Cargas							
Máquinas Virtuais Possuindo Somente uma Prioridade							
1,3%	**28,6%**	2,3%	2,3%	21,1%	5,2%	12,8%	11,2%
2,9%	2,9%	1,8%	2,9%	1,3%	3,4%	15,1%	
Tempo Médio de Processamento das Cargas — Alta Prioridade							
Máquinas Virtuais Possuindo Somente uma Prioridade							
100,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%
0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	
Tempo Médio de Processamento das Cargas — Prioridade Normal							
Máquinas Virtuais Possuindo Somente uma Prioridade							
1,3%	**28,6%**	2,3%	2,3%	21,1%	5,2%	12,8%	11,2%
2,9%	2,9%	1,8%	2,9%	1,3%	3,4%	15,1%	
Tempo Médio de Processamento das Cargas — Baixa Prioridade							
Máquinas Virtuais Possuindo Somente uma Prioridade							
100,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%
0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	
Tempo Médio para Conclusão das Cargas							
Máquinas Virtuais Possuindo Somente uma Prioridade							
0,0%	4,2%	**15,6%**	4,2%	9,1%	9,1%	3,4%	5,7%
8,1%	9,4%	5,7%	8,6%	8,9%	8,1%	48,7%	
Tempo Médio para Conclusão das Cargas — Alta Prioridade							
Máquinas Virtuais Possuindo Somente uma Prioridade							
100,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%
0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	
Tempo Médio para Conclusão das Cargas — Prioridade Normal							
Máquinas Virtuais Possuindo Somente uma Prioridade							
0,0%	4,2%	**15,6%**	4,2%	9,1%	9,1%	3,4%	5,7%
8,1%	9,4%	5,7%	8,6%	8,9%	8,1%	48,7%	

RND	RR	FA	HU	HR	MPD	LA	BALA
TEAp	TEApf	TEAb	TEAbf	TEAe	TEAef	TEA	
Tempo Médio para Conclusão das Cargas — Baixa Prioridade							
Máquinas Virtuais Possuindo Somente uma Prioridade							
100,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%
0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	
Consumo de Energia Total Médio							
Máquinas Virtuais Possuindo Múltiplas Prioridades							
0,0%	0,0%	13,8%	0,5%	**16,9%**	5,5%	3,4%	5,7%
4,9%	6,5%	5,7%	7,3%	13,8%	15,9%	54,2%	
Makespan Médio							
Máquinas Virtuais Possuindo Múltiplas Prioridades							
0,3%	0,8%	9,1%	9,9%	3,6%	0,5%	3,1%	4,2%
11,7%	10,4%	9,6%	**13,5%**	11,2%	12,0%	68,5%	
Tempo Médio de Processamento das Cargas							
Máquinas Virtuais Possuindo Múltiplas Prioridades							
7,6%	**28,6%**	3,1%	2,9%	21,9%	2,9%	11,2%	10,2%
1,0%	2,3%	1,3%	3,1%	2,1%	1,8%	11,7%	
Tempo Médio de Processamento das Cargas — Alta Prioridade							
Máquinas Virtuais Possuindo Múltiplas Prioridades							
5,7%	**21,1%**	2,6%	2,1%	15,4%	2,6%	10,7%	14,3%
4,4%	4,2%	3,9%	4,2%	5,2%	3,6%	25,5%	
Tempo Médio de Processamento das Cargas — Prioridade Normal							
Máquinas Virtuais Possuindo Múltiplas Prioridades							
7,0%	**27,1%**	2,9%	4,2%	16,1%	6,8%	9,4%	9,1%
3,1%	3,9%	2,3%	1,8%	3,4%	2,9%	17,4%	
Tempo Médio de Processamento das Cargas — Baixa Prioridade							
Máquinas Virtuais Possuindo Múltiplas Prioridades							
10,7%	**26,8%**	1,0%	2,3%	18,2%	7,8%	12,2%	9,4%
0,5%	2,6%	2,9%	2,1%	2,3%	1,0%	11,5%	
Tempo Médio para Conclusão das Cargas							
Máquinas Virtuais Possuindo Múltiplas Prioridades							
0,0%	4,4%	**18,8%**	12,5%	4,9%	7,8%	5,2%	7,3%
4,4%	6,8%	6,8%	8,1%	6,0%	7,0%	39,1%	
Tempo Médio para Conclusão das Cargas — Alta Prioridade							
Máquinas Virtuais Possuindo Múltiplas Prioridades							
1,3%	2,9%	11,2%	10,2%	9,9%	8,9%	3,6%	6,0%
7,3%	**12,2%**	4,7%	7,6%	9,6%	4,7%	46,1%	

RND	RR	FA	HU	HR	MPD	LA	BALA
TEAp	TEApf	TEAb	TEAbf	TEAe	TEAef	TEA	
Tempo Médio para Conclusão das Cargas — Prioridade Normal							
Máquinas Virtuais Possuindo Múltiplas Prioridades							
0,0%	5,5%	**16,7%**	13,5%	3,4%	7,3%	6,5%	7,3%
6,3%	7,8%	6,8%	7,6%	5,5%	6,0%	39,8%	
Tempo Médio para Conclusão das Cargas — Baixa Prioridade							
Máquinas Virtuais Possuindo Múltiplas Prioridades							
0,3%	6,3%	**18,2%**	13,0%	4,9%	5,5%	5,5%	6,8%
6,5%	7,6%	6,5%	6,5%	6,5%	6,0%	39,6%	

Estudando exclusivamente o consumo de energia total médio da nuvem, analisamos os valores relativos ao percentual do número de vezes em que cada algoritmo foi considerado o melhor para os cenários avaliados:

- `BALA` foi o melhor em 5,2% dos cenários avaliados;
- `FA` foi o melhor em 11,1% dos cenários avaliados;
- `HR` foi o melhor em 15,0% dos cenários avaliados;
- `HU` foi o melhor em 0,4% dos cenários avaliados;
- `LA` foi o melhor em 3,9% dos cenários avaliados;
- `MPD` foi o melhor em 4,4% dos cenários avaliados;
- `TEAb` foi o melhor em 6,4% dos cenários avaliados;
- `TEAbf` foi o melhor em 7,7% dos cenários avaliados;
- `TEAe` foi o melhor em 15,5% dos cenários avaliados;
- `TEAef` foi o melhor em 15,4% dos cenários avaliados;
- `TEAp` foi o melhor em 6,9% dos cenários avaliados;
- `TEApf` foi o melhor em 8,2% dos cenários avaliados;
- Portanto, o `TEA` foi o melhor em 60,0% dos cenários avaliados para este critério.

Considerando exclusivamente o *makespan* médio da nuvem, analisamos os valores relativos do percentual do número de vezes em que cada algoritmo foi considerado o melhor para os cenários avaliados:

- `BALA` foi o melhor em 3,6% dos cenários avaliados;
- `FA` foi o melhor em 5,5% dos cenários avaliados;

- HR foi o melhor em 2,7% dos cenários avaliados;
- HU foi o melhor em 6,0% dos cenários avaliados;
- LA foi o melhor em 2,5% dos cenários avaliados;
- MPD foi o melhor em 1,0% dos cenários avaliados;
- RND foi o melhor em 0,1% dos cenários avaliados;
- RR foi o melhor em 0,9% dos cenários avaliados;
- TEAb foi o melhor em 10,9% dos cenários avaliados;
- TEAbf foi o melhor em 15,5% dos cenários avaliados;
- TEAe foi o melhor em 12,8% dos cenários avaliados;
- TEAef foi o melhor em 12,8% dos cenários avaliados;
- TEAp foi o melhor em 13,0% dos cenários avaliados;
- TEApf foi o melhor em 12,6% dos cenários avaliados;
- Portanto, o TEA foi o melhor em 77,6% dos cenários avaliados para este critério.

Em uma reflexão exclusiva sobre o tempo médio de processamento das cargas de trabalho na nuvem, observamos os valores relativos do percentual do número de vezes em que cada algoritmo foi considerado o melhor para os cenários avaliados:

- BALA foi o melhor em 10,7% dos cenários avaliados;
- FA foi o melhor em 2,7% dos cenários avaliados;
- HR foi o melhor em 21,5% dos cenários avaliados;
- HU foi o melhor em 2,6% dos cenários avaliados;
- LA foi o melhor em 12,0% dos cenários avaliados;
- MPD foi o melhor em 4,0% dos cenários avaliados;
- RND foi o melhor em 4,4% dos cenários avaliados;
- RR foi o melhor em 28,6% dos cenários avaliados;
- TEAb foi o melhor em 1,6% dos cenários avaliados;
- TEAbf foi o melhor em 3,0% dos cenários avaliados;
- TEAe foi o melhor em 1,7% dos cenários avaliados;
- TEAef foi o melhor em 2,6% dos cenários avaliados;

- TEAp foi o melhor em 2,0% dos cenários avaliados;
- TEApf foi o melhor em 2,6% dos cenários avaliados;
- Portanto, o TEA foi o melhor em 13,4% dos cenários avaliados para este critério.

Em uma reflexão exclusiva sobre o tempo médio de processamento das cargas de trabalho de alta prioridade na nuvem, observamos os valores relativos do percentual do número de vezes em que cada algoritmo foi considerado o melhor para os cenários avaliados:

- BALA foi o melhor em 7,2% dos cenários avaliados;
- FA foi o melhor em 1,3% dos cenários avaliados;
- HR foi o melhor em 7,7% dos cenários avaliados;
- HU foi o melhor em 1,0% dos cenários avaliados;
- LA foi o melhor em 5,3% dos cenários avaliados;
- MPD foi o melhor em 1,3% dos cenários avaliados;
- RND foi o melhor em 52,9% dos cenários avaliados;
- RR foi o melhor em 10,5% dos cenários avaliados;
- TEAb foi o melhor em 2,0% dos cenários avaliados;
- TEAbf foi o melhor em 2,1% dos cenários avaliados;
- TEAe foi o melhor em 2,6% dos cenários avaliados;
- TEAef foi o melhor em 1,8% dos cenários avaliados;
- TEAp foi o melhor em 2,2% dos cenários avaliados;
- TEApf foi o melhor em 2,1% dos cenários avaliados;
- Portanto, o TEA foi o melhor em 12,8% dos cenários avaliados para este critério.

Em uma reflexão exclusiva sobre o tempo médio de processamento das cargas de trabalho de prioridade normal na nuvem, observamos os valores relativos do percentual do número de vezes em que cada algoritmo foi considerado o melhor para os cenários avaliados:

- BALA foi o melhor em 10,2% dos cenários avaliados;
- FA foi o melhor em 2,6% dos cenários avaliados;
- HR foi o melhor em 18,6% dos cenários avaliados;
- HU foi o melhor em 3,3% dos cenários avaliados;

- `LA` foi o melhor em 11,1% dos cenários avaliados;
- `MPD` foi o melhor em 6,0% dos cenários avaliados;
- `RND` foi o melhor em 4,2% dos cenários avaliados;
- `RR` foi o melhor em 27,9% dos cenários avaliados;
- `TEAb` foi o melhor em 2,1% dos cenários avaliados;
- `TEAbf` foi o melhor em 2,3% dos cenários avaliados;
- `TEAe` foi o melhor em 2,3% dos cenários avaliados;
- `TEAef` foi o melhor em 3,1% dos cenários avaliados;
- `TEAp` foi o melhor em 3,0% dos cenários avaliados;
- `TEApf` foi o melhor em 3,4% dos cenários avaliados;
- Portanto, o `TEA` foi o melhor em 16,3% dos cenários avaliados para este critério.

Em uma reflexão exclusiva sobre o tempo médio de processamento das cargas de trabalho de baixa prioridade na nuvem, observamos os valores relativos do percentual do número de vezes em que cada algoritmo foi considerado o melhor para os cenários avaliados:

- `BALA` foi o melhor em 4,7% dos cenários avaliados;
- `FA` foi o melhor em 0,5% dos cenários avaliados;
- `HR` foi o melhor em 9,1% dos cenários avaliados;
- `HU` foi o melhor em 1,2% dos cenários avaliados;
- `LA` foi o melhor em 6,1% dos cenários avaliados;
- `MPD` foi o melhor em 3,9% dos cenários avaliados;
- `RND` foi o melhor em 55,3% dos cenários avaliados;
- `RR` foi o melhor em 13,4% dos cenários avaliados;
- `TEAb` foi o melhor em 1,4% dos cenários avaliados;
- `TEAbf` foi o melhor em 1,0% dos cenários avaliados;
- `TEAe` foi o melhor em 1,2% dos cenários avaliados;
- `TEAef` foi o melhor em 0,5% dos cenários avaliados;
- `TEAp` foi o melhor em 0,3% dos cenários avaliados;

- TEApf foi o melhor em 1,3% dos cenários avaliados;
- Portanto, o TEA foi o melhor em 5,7% dos cenários avaliados para este critério.

Considerando exclusivamente o tempo médio para conclusão das cargas de trabalho na nuvem, observamos os valores relativos do percentual do número de vezes em que cada algoritmo foi considerado o melhor para os cenários avaliados:

- BALA foi o melhor em 6,5% dos cenários avaliados;
- FA foi o melhor em 17,2% dos cenários avaliados;
- HR foi o melhor em 7,0% dos cenários avaliados;
- HU foi o melhor em 8,3% dos cenários avaliados;
- LA foi o melhor em 4,3% dos cenários avaliados;
- MPD foi o melhor em 8,5% dos cenários avaliados;
- RR foi o melhor em 4,3% dos cenários avaliados;
- TEAb foi o melhor em 6,3% dos cenários avaliados;
- TEAbf foi o melhor em 8,3% dos cenários avaliados;
- TEAe foi o melhor em 7,4% dos cenários avaliados;
- TEAef foi o melhor em 7,6% dos cenários avaliados;
- TEAp foi o melhor em 6,3% dos cenários avaliados;
- TEApf foi o melhor em 8,1% dos cenários avaliados;
- Portanto, o TEA foi o melhor em 43,9% dos cenários avaliados para este critério.

Considerando exclusivamente o tempo médio para conclusão das cargas de trabalho de alta prioridade na nuvem, observamos os valores relativos do percentual do número de vezes em que cada algoritmo foi considerado o melhor para os cenários avaliados:

- BALA foi o melhor em 3,0% dos cenários avaliados;
- FA foi o melhor em 5,6% dos cenários avaliados;
- HR foi o melhor em 4,9% dos cenários avaliados;
- HU foi o melhor em 5,1% dos cenários avaliados;
- LA foi o melhor em 1,8% dos cenários avaliados;
- MPD foi o melhor em 4,4% dos cenários avaliados;
- RND foi o melhor em 50,7% dos cenários avaliados;

- RR foi o melhor em 1,4% dos cenários avaliados;
- TEAb foi o melhor em 2,3% dos cenários avaliados;
- TEAbf foi o melhor em 3,8% dos cenários avaliados;
- TEAe foi o melhor em 4,8% dos cenários avaliados;
- TEAef foi o melhor em 2,3% dos cenários avaliados;
- TEAp foi o melhor em 3,6% dos cenários avaliados;
- TEApf foi o melhor em 6,1% dos cenários avaliados;
- Portanto, o TEA foi o melhor em 23,0% dos cenários avaliados para este critério.

Considerando exclusivamente o tempo médio para conclusão das cargas de trabalho de prioridade normal na nuvem, observamos os valores relativos do percentual do número de vezes em que cada algoritmo foi considerado o melhor para os cenários avaliados:

- BALA foi o melhor em 6,5% dos cenários avaliados;
- FA foi o melhor em 16,1% dos cenários avaliados;
- HR foi o melhor em 6,3% dos cenários avaliados;
- HU foi o melhor em 8,9% dos cenários avaliados;
- LA foi o melhor em 4,9% dos cenários avaliados;
- MPD foi o melhor em 8,2% dos cenários avaliados;
- RR foi o melhor em 4,8% dos cenários avaliados;
- TEAb foi o melhor em 6,3% dos cenários avaliados;
- TEAbf foi o melhor em 8,1% dos cenários avaliados;
- TEAe foi o melhor em 7,2% dos cenários avaliados;
- TEAef foi o melhor em 7,0% dos cenários avaliados;
- TEAp foi o melhor em 7,2% dos cenários avaliados;
- TEApf foi o melhor em 8,6% dos cenários avaliados;
- Portanto, o TEA foi o melhor em 44,3% dos cenários avaliados para este critério.

Considerando exclusivamente o tempo médio para conclusão das cargas de trabalho de baixa prioridade na nuvem, observamos os valores relativos do percentual do número de vezes em que cada algoritmo foi considerado o melhor para os cenários avaliados:

- BALA foi o melhor em 3,4% dos cenários avaliados;

- `FA` foi o melhor em 9,1% dos cenários avaliados;
- `HR` foi o melhor em 2,5% dos cenários avaliados;
- `HU` foi o melhor em 6,5% dos cenários avaliados;
- `LA` foi o melhor em 2,7% dos cenários avaliados;
- `MPD` foi o melhor em 2,7% dos cenários avaliados;
- `RND` foi o melhor em 50,1% dos cenários avaliados;
- `RR` foi o melhor em 3,1% dos cenários avaliados;
- `TEAb` foi o melhor em 3,3% dos cenários avaliados;
- `TEAbf` foi o melhor em 3,3% dos cenários avaliados;
- `TEAe` foi o melhor em 3,3% dos cenários avaliados;
- `TEAef` foi o melhor em 3,0% dos cenários avaliados;
- `TEAp` foi o melhor em 3,3% dos cenários avaliados;
- `TEApf` foi o melhor em 3,8% dos cenários avaliados;
- Portanto, o `TEA` foi o melhor em 19,8% dos cenários avaliados para este critério.

Considerando o `TEA` e comparando suas diferentes implementações entre si e com os melhores algoritmos nos critérios estudados, observamos que:

- Considerando exclusivamente a média do consumo de energia, a melhor versão do `TEA` foi a `TEAe` (em 15,5% dos casos);
- Considerando exclusivamente o *makespan* médio, a melhor versão do `TEA` foi a `TEAbf` (em 15,5% dos casos);
- Considerando exclusivamente a média do tempo de processamento de cargas de trabalho, a melhor versão do `TEA` foi a `TEAbf` (em 3,0% dos casos);
- Considerando exclusivamente a média do tempo de processamento de cargas de trabalho com alta prioridade, a melhor versão do `TEA` foi a `TEAe` (em 2,6% dos casos);
- Considerando exclusivamente a média do tempo de processamento de cargas de trabalho com prioridade normal, a melhor versão do `TEA` foi a `TEApf` (em 3,4% dos casos);
- Considerando exclusivamente a média do tempo de processamento de cargas de trabalho com baixa prioridade, a melhor versão do `TEA` foi a `TEAb` (em 1,4% dos casos);

- Considerando exclusivamente a média do tempo para conclusão de cargas de trabalho, a melhor versão do `TEA` foi a `TEAbf` (em 8,3% dos casos);
- Considerando exclusivamente a média do tempo para conclusão de cargas de trabalho com alta prioridade, a melhor versão do `TEA` foi a `TEApf` (em 6,1% dos casos);
- Considerando exclusivamente a média do tempo para conclusão de cargas de trabalho com prioridade normal, a melhor versão do `TEA` foi a `TEApf` (em 8,6% dos casos);
- Considerando exclusivamente a média do tempo para conclusão de cargas de trabalho com baixa prioridade, a melhor versão do `TEA` foi a `TEApf` (em 3,8% dos casos).
- Analisando a média do consumo de energia, considerando os intervalos de confiança em 95% de todos os algoritmos, a versão do `TEA` que apresentou os melhores resultados foi `TEAe` (com o `TEA` sendo melhor em 63,0% dos casos entre todos os algoritmos avaliados);
- Analisando o *makespan* médio, considerando os intervalos de confiança em 95% de todos os algoritmos, a versão do `TEA` que apresentou os melhores resultados foi `TEAb` (com o `TEA` sendo melhor em 80,6% dos casos entre todos os algoritmos avaliados);
- Analisando a média do tempo de processamento de cargas de trabalho, considerando os intervalos de confiança em 95% de todos os algoritmos, a versão do `TEA` que apresentou os melhores resultados foi `TEAbf` (com o `TEA` sendo melhor em 28,4% dos casos entre todos os algoritmos avaliados);
- Analisando a média do tempo de processamento de cargas de trabalho de alta prioridade, considerando os intervalos de confiança em 95% de todos os algoritmos, a versão do `TEA` que apresentou os melhores resultados foi `TEAp` (com o `TEA` sendo melhor em 23,7% dos casos entre todos os algoritmos avaliados);
- Analisando a média do tempo de processamento de cargas de trabalho de prioridade normal, considerando os intervalos de confiança em 95% de todos os algoritmos, a versão do `TEA` que apresentou os melhores resultados foi `TEAbf` (com o `TEA` sendo melhor em 35,2% dos casos entre todos os algoritmos avaliados);
- Analisando a média do tempo de processamento de cargas de trabalho de baixa prioridade, considerando os intervalos de confiança em 95% de todos os algoritmos, a versão do `TEA` que apresentou os melhores resultados foi `TEAp` (com o `TEA` sendo melhor em 16,1% dos casos entre todos os algoritmos avaliados);
- Analisando a média do tempo para conclusão de cargas de trabalho, considerando os intervalos de confiança em 95% de todos os algoritmos, a versão do `TEA` que apresentou os melhores resultados foi `TEAp` (com o `TEA` sendo melhor em 45,4% dos casos entre todos os algoritmos avaliados);

- Analisando a média do tempo para conclusão de cargas de trabalho de alta prioridade, considerando os intervalos de confiança em 95% de todos os algoritmos, a versão do `TEA` que apresentou os melhores resultados foi `TEAp` (com o `TEA` sendo melhor em 25,4% dos casos entre todos os algoritmos avaliados);
- Analisando a média do tempo para conclusão de cargas de trabalho de prioridade normal, considerando os intervalos de confiança em 95% de todos os algoritmos, a versão do `TEA` que apresentou os melhores resultados foi `TEAp` (com o `TEA` sendo melhor em 46,0% dos casos entre todos os algoritmos avaliados);
- Analisando a média do tempo para conclusão de cargas de trabalho de baixa prioridade, considerando os intervalos de confiança em 95% de todos os algoritmos, a versão do `TEA` que apresentou os melhores resultados foi `TEAp` (com o `TEA` sendo melhor em 21,0% dos casos entre todos os algoritmos avaliados).

6.5 Comentários Finais

Neste capítulo expandimos nosso foco para prover eficiência em energia considerando não somente os elementos de borda, mas também trazendo ao algoritmo a ciência da topologia em si.

Empregando os conhecimentos obtidos com os estudos anteriores, propomos um novo e escalável algoritmo de escalonamento de máquinas virtuais, o `TEA`. Este algoritmo foi projetado para atuar em duas frentes: (a) nas atualizações do *broker*, realizando um pré-processamento da ordem na qual as máquinas virtuais serão tratadas, de modo estratégico para prover eficiência em energia sem prejudicar as prioridades; e (b) na escolha do servidor para hospedar as máquinas virtuais, objetivando prover eficiência em energia através de cinco critérios: (i) menor diferença em potência consumida antes e após a alocação da máquina virtual; servidor com maior eficiência em energia com relação (ii) ao MIPS, (iii) à quantidade de RAM, (iv) à largura de banda; e (v) rota mais eficiente em energia entre o sistema final no qual a máquina virtual se encontra e o servidor de destino.

Este algoritmo também atua com duas possibilidades de personalização: (i) limitação do número de servidores ligados — os mais eficientes em energia — para cargas não prioritárias e (ii) uso de um estimador de tempo restante de máquinas virtuais para tentar otimizar o processo de escalonamento. Com base nestas personalizações, oferecemos e testamos seis sabores do algoritmo: quanto à limitação do número de servidores para uso geral ligados oferecemos (i) uma versão econômica `TEAe`, com o limite de 50%; (ii) uma versão balanceada `TEAb`, com limite de 75%; e (iii) uma versão focada em desempenho `TEAp`, que admite ligar todos os servidores. Além destes, oferecemos mais três sabores, que tentam estimar o tempo restante das máquinas virtuais com um algoritmo estatístico de modo a tentar otimizar o processo de escalonamento, a saber, respectivamente, `TEAef`, `TEAbf` e `TEApf`.

Nossos testes mostraram que todos os sabores do `TEA` se comportaram de forma satisfatória em todos os cenários, com ganhos notáveis em *data centers* geodistribuídos, heterogêneos e com topologias constituídas por uma numerosa quantidade de comutado-

res de pacotes ou cujas rotas possam passar por um grande número de comutadores de pacotes. Vale aqui ressaltar que o principal objetivo da atuação do `TEA` é na minimização do consumo de energia. No entanto, do ponto de vista de custos operacionais, em especial no caso de geodistribuição, taxas inerentes à manutenção de *data centers* — e.g. interligação — também podem ser passíveis de análise pelo administrador da nuvem.

A melhor versão do `TEA` para economia de energia foi a `TEAe` e para o *makespan* foi a `TEAb`. Quanto aos tempos de processamento de cargas independente de prioridade a melhor versão foi a `TEAbf`, enquanto com prioridades alta, média e baixa foram, respectivamente, `TEAp`, `TEAbf` e `TEAp`. Sobre o tempo para conclusão das cargas, para todos os casos a melhor versão foi a `TEAp`.

Considerando o intervalo de confiança, o `TEA` se mostrou melhor do que todos os algoritmos em 63% dos casos de eficiência em energia.

Os resultados obtidos por simulação mostraram que, tanto nos casos da análise exclusiva de média, quanto considerando os intervalos de confiança, o recurso de estimativa do tempo para finalização de máquinas virtuais utilizado `f` gerou os melhores, porém limitados, resultados nos tempos de processamento de cargas, independente de prioridade, sugerindo que este recurso pode trazer melhoras neste sentido. Apesar disso, não podemos tecer a mesma afirmação acerca do *makespan* e do tempo para conclusão de cargas de trabalho.

Portanto, podemos dizer que o `TEA` trouxe ganhos consideráveis para o processo de escalonamento ciente de energia, corroborando a importância de se olhar para o núcleo da rede no processo de escalonamento de máquinas virtuais em nuvens computacionais.

Capítulo 7

Conclusão

Neste trabalho observamos as consequências — na forma de heterogeneidade — da rápida evolução dos hardwares usados na computação em nuvem, apresentando nossas contribuições para maximizar a eficiência em energia e a redução da poluição gerada por *data centers* através do estudo e propostas de sistemas de escalonamento ciente de energia de máquinas virtuais que operam em nuvem.

Entregamos, como primeiras contribuições, o resultado de um extenso estudo acerca dos impactos das redes de alta velocidade em *data centers* que operam em nuvem, que continuam emergindo conforme a consistente evolução da tecnologia *Ethernet*. Vimos que as redes de alta velocidade podem impactar de forma significativa o processo de escalonamento de máquinas virtuais, em especial no que diz respeito sobre a submissão de máquinas virtuais — possibilitando instanciações mais rápidas destas em cenários de armazenamento distribuídos de imagens — e a migração de máquinas virtuais — tornando-as mais rápidas. Vimos que os impactos dependem do algoritmo de escalonamento empregado e que algoritmos que utilizam menos a rede — e.g., os que realizam menos migrações — são menos beneficiados. Notamos que as redes de alta velocidade podem trazer um custo energético associado, mas que os ganhos obtidos com os impactos destas podem compensar. Norteamos sobre como calcular se compensa ou não a substituição das redes de alta velocidade nessas situações. Apresentamos um modelo empírico para estimar o consumo de energia de *data centers* com tais redes, mostrando o comportamento dos *data centers* sob redes de alta velocidade. Vimos que o comportamento do consumo de energia, do *makespan* e do tempo de execução de cargas é de decaimento exponencial em função da velocidade da rede — quando admitimos a potência constante da rede — e que existe um limite superior após o qual o aumento da velocidade da rede não trás benefícios substanciais — para os cenários estudados benefícios muito pequenos foram observados para redes com velocidades acima de 100 Gbps. Observamos que o número de migrações em redes de alta velocidade tendem a aumentar até um limite de velocidade e, após este, tal número pode sofrer grande queda, mas isso é dependente do algoritmo de escalonamento utilizado e uma possível hipótese para explicar isso é que redes de alta velocidade possibilitam disponibilização mais rápida de servidores e, com tal disponibilização, os algoritmos de escalonamento são capazes de realizar alocações otimizadas.

Notando os impactos da evolução dos equipamentos de hardware em geral, inclusive redes, observamos o potencial surgimento de *data centers* cada vez mais heterogêneos, in-

clusive em largura de banda. De posse dos aprendizados obtidos com os estudos anteriores e a experiência prévia em algoritmos de escalonamento cientes de energia, propomos um novo algoritmo, ciente de energia, que considerava diversos aspectos dos sistemas finais, e como contribuição passou a ponderar também a largura de banda. Tal algoritmo, o `BALA`, atua através de sucessivos critérios de desempate de características entre os servidores de uma nuvem para definir qual é o melhor para uma dada máquina virtual ingressante, a saber: (i) a maior eficiência em energia, (ii) menor diferença do consumo de potência estimada antes e após a alocação da máquina virtual em questão, (iii) servidor com maior utilização, (iv) servidor com maior capacidade de processamento e (v) servidor com maior largura de banda. Originalmente este algoritmo foi projetado para trabalhar juntamente a algoritmos de provisionamento de largura de banda para máquinas virtuais, executados por servidores, mas também apresentamos posteriormente este algoritmo atuando sem tal simbiose. Apresentamos um exemplo de algoritmo de reserva de largura de banda para lidar com o tais reservas, o `+E`, que tem por objetivo fazer reservas de largura de banda não utilizada igualitariamente entre todas as máquinas virtuais alocadas em um dado servidor, para acelerar o processo de migração de máquinas virtuais. Mostramos também como a reserva adicional de banda pode ser melhorada com duas ideias de algoritmos: (i) um algoritmo de reserva de banda diferenciado, complexo, customizável para atender às solicitações do administrador da nuvem, ou (ii) um sub-tipo do algoritmo (i), adaptativo, capaz de resolver uma limitação vista no `+E` que é, em alguns casos, não conseguir fazer reserva de toda a largura de banda adicional para a migração de máquinas virtuais. Quanto ao consumo de energia e ao *makespan*, mostramos que o `BALA` é uma opção aceitável para *data centers* homogêneos, mas dentre os algoritmos estudados se mostrou a melhor opção para *data centers* heterogêneos. Vimos que, com relação ao consumo de energia, nos casos em que o `BALA` se mostrou pior do que os demais — notavelmente em *data centers* homogêneos — esta piora foi em média de 10%; mas nos cenários em que ele foi melhor, o desempenho foi superior a 43%. No caso do *makespan*, esta piora e melhora giraram em, respectivamente, 1% e 7%.

Em outra contribuição, aumentamos a abrangência de nosso foco para considerar não somente os servidores, mas também o núcleo da rede, em diversas topologias e os impactos das redes de alta velocidade. Neste sentido, propomos o nosso último algoritmo de escalonamento, o `TEA`, baseado em dois pilares: (a) em toda a atualização do *broker*, realizar um pré-processamento estratégico da ordem das máquinas virtuais por este atendidas; e (b) o algoritmo para encontrar servidores para máquinas virtuais, ciente de energia e de topologia de rede. O pré-processamento da ordem das máquinas virtuais visou dar preferência máquinas virtuais (i) que necessitam de maior prioridade, (ii) que apresentam maior eficiência em energia — considerando MIPS, RAM e largura de banda — (iii) a máquina virtual que apresenta maior tempo restante estimado para ser finalizada, (iv) solicitação de recursos pela máquina virtual — RAM e MIPS — (v) tamanho do arquivo de imagem da máquina virtual, (vi) MIPS efetivo da máquina virtual, (vii) quantidade de carga já processada pela máquina virtual. Quanto ao algoritmo de determinação de servidor para receber uma máquina virtual, os princípios foram: (i) menor diferença de consumo de potência estimada antes e após alocar a máquina virtual, (ii) eficiência em energia considerando a capacidade de processamento do servidor, (iii) eficiência em energia

considerando a quantidade de RAM do servidor, (iv), eficiência em energia considerando a largura de banda do servidor, (v) eficiência em energia considerando a rota entre o sistema final onde máquina virtual se encontra e o servidor de destino. O `TEA` se mostrou capaz de atuar em um grande número de cenários, como em nuvens constituídas por *data centers* homogêneos ou heterogêneos, geodistribuídos ou não, com topologias arbitrárias, com taxas de transmissão arbitrárias de quaisquer elementos de núcleo e de borda, com alguns SLAs, com ou sem o recurso de migração de máquinas virtuais e escalável. Os resultados obtidos por simulação, para uma enorme gama de cenários, mostrou a eficiência do algoritmo desenvolvido, sendo capaz de, quando comparado a diversos outros algoritmos eficientes em energia, ser a melhor opção em (i) mais da metade dos cenários quanto ao consumo de energia, (ii) redução do *makespan* em 3/4 dos casos, (iii) redução do tempo de processamento de cargas de trabalho em um décimo dos casos e (iv) redução do tempo médio para conclusão das cargas de trabalho em um décimo dos casos; não sendo inaceitavelmente mais lento que os demais em caso algum. Comparado aos demais algoritmos, observamos ganhos notáveis em *data centers* geodistribuídos, heterogêneos e com topologias constituídas por um número grande de comutadores de pacotes ou cujas rotas possam passar por um elevado número de comutadores de pacotes.

Além dos trabalhos apresentados, deixamos como legado também o *SinergyCloud*, um *framework open-source* de simulação para nuvens, projetado com alta granularidade, e considerando em especial a rede. O *SinergyCloud* foi desenvolvido para ser facilmente usado em um grande número de cenários de nuvens, provendo suporte a um grande número de características e recursos de nuvens, como geodistribuição de *data centers*, topologias arbitrárias e migrações de máquinas virtuais, possibilitando aos pesquisadores testar detalhadamente suas soluções com ênfase em consumo de energia IaaS, em especial com relação ao processo de escalonamento. O *toolkit* de simulação proposto é facilmente extensível, com um projeto modular que garante a fácil adição de novos módulos, além de permitir usuários de configurar rapidamente suas simulações com uma arquitetura de fácil uso. Como visto neste trabalho, o *SinergyCloud* também provê um grande espectro de elementos de modelagem e percepção do processo de simulação, enquanto apresentando boas métricas de desempenho, mesmo em simulações de larga escala.

Como propostas de trabalhos futuros, pretendemos trabalhar com outros modelos de distribuição de cargas além do *bag-of-tasks*, como por exemplo, *workflows* e com *traces* de organizações; trabalhar com o intuito de maximizar a eficiência de estruturas que consomem energia externas ao núcleo dos *data centers*, em especial às relacionadas com resfriamento, otimizando o *PUE*; considerar geodistribuição com *data centers* com diferentes fontes de energia (ex: renováveis e não renováveis), fusos horários e zonas climáticas; objetivar a disposição próxima de máquinas virtuais com elevada inter-comunicação; e realizar otimizações no contexto de *Storage Area Networks*.

Referências Bibliográficas

[1] Green Grid: 7x24 Change International, Ashrae, Silicon Valley Leadership Group, Program, Energy S. Epa, Gbc, and 2010. Recommendations for measuring and reporting overall data center efficiency. Technical report, Uptime Institute, 2010.

[2] P. Agrawal and S. Rao. Energy-aware scheduling of distributed systems. *Automation Science and Engineering, IEEE Transactions on*, 11(4):1163–1175, Oct 2014.

[3] Amazon. Amazon elastic compute cloud (Amazon EC2). http://aws.amazon.com/ec2/, Acessado em 15/01/2018.

[4] Inc. Amazon Web Services. Amazon ec2 instance types. https://aws.amazon.com/ec2/instance-types/, Acessado em 15/01/2018.

[5] AMD. Cool n quiet technology installation guide for amd athlon 64 processor based systems. http://www.amd.com/Documents/Cool_N_Quiet_Installation_Guide3.pdf, 2010. Acessado em 15/01/2018.

[6] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, et al. A view of cloud computing. *Communications of the ACM*, 53(4):50–58, 2010.

[7] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David A. Patterson, Ariel Rabkin, Ion Stoica, and Matei Zaharia. Above the Clouds: A Berkeley View of Cloud Computing. Technical Report UCB/EECS-2009-28, EECS Department, University of California, Berkeley, Feb 2009.

[8] Amazon AWS. Amazon web services & intel, 2017.

[9] Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauer, Ian Pratt, and Andrew Warfield. Xen and the art of virtualization. *SIGOPS Oper. Syst. Rev.*, 37:164–177, October 2003.

[10] Anton Beloglazov, Jemal Abawajy, and Rajkumar Buyya. Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. *Future Generation Computer Systems*, 28(5):755 – 768, 2012. Special Section: Energy efficiency in large-scale distributed systems.

[11] Anton Beloglazov and Rajkumar Buyya. Energy efficient resource management in virtualized cloud data centers. In *Proceedings of the 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*, CCGRID '10, pages 826–831, Washington, DC, USA, 2010. IEEE Computer Society.

[12] Andreas Bergen, Ronald Desmarais, Sudhakar Ganti, and Ulrike Stege. Towards software-adaptive green computing based on server power consumption. In *Proceedings of the 3rd International Workshop on Green and Sustainable Software*, GREENS 2014, pages 9–16, New York, NY, USA, 2014. ACM.

[13] A Berl, E Gelenbe, M Di Girolamo, G Giuliani, H De Meer, M Q Dang, and K Pentikousis. Energy-efficient cloud computing. *The Computer Journal*, 53(7):1045–1051, 2009.

[14] Luciano Bertini, Julius C. B. Leite, and Daniel Mossé. Power optimization for dynamic configuration in heterogeneous web server clusters. *J. Syst. Softw.*, 83:585–598, April 2010.

[15] Luiz Bezerra. Cloud computing. Tecnologia e Gestão. https://tecnologiaegestao.wordpress.com/2010/06/28/cloud-computing/, 2010. Acessado em 15/01/2018.

[16] Walter Binder and Niranjan Suri. Green computing: Energy consumption optimized service hosting. In *Proceedings of the 35th Conference on Current Trends in Theory and Practice of Computer Science*, SOFSEM '09, pages 117–128, Berlin, Heidelberg, 2009. Springer-Verlag.

[17] Bui, Sang T. (Irvine, CA, US). Method and apparatus for performing wake on lan power management, Maio 2006. Patent Number 7047428.

[18] Marc Bux and Ulf Leser. Dynamiccloudsim: Simulating heterogeneity in computational clouds. *Future Generation Computer Systems*, 46:85 – 99, 2015.

[19] R. Buyya, R. Ranjan, and R.N. Calheiros. Modeling and simulation of scalable cloud computing environments and the cloudsim toolkit: Challenges and opportunities. In *Intl. Conf. on High Performance Computing & Simulation, 2009. HPCS'09.*, pages 1–11, 2009.

[20] Rajkumar Buyya and Manzur Murshed. Gridsim: A toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing. *Concurrency and computation: practice and experience*, 14(13-15):1175–1220, 2002.

[21] Rajkumar Buyya, Chee Shin Yeo, Srikumar Venugopal, James Broberg, and Ivona Brandic. Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Gener. Comput. Syst.*, 25:599–616, June 2009.

[22] Rodrigo N Calheiros, Rajiv Ranjan, Anton Beloglazov, César AF De Rose, and Rajkumar Buyya. Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software: Practice and experience*, 41(1):23–50, 2011.

[23] Weiwei Chen and Ewa Deelman. Workflowsim: A toolkit for simulating scientific workflows in distributed environments. In *Proceedings of the 2012 IEEE 8th International Conference on E-Science (e-Science)*, E-SCIENCE '12, pages 1–8, Washington, DC, USA, 2012. IEEE Computer Society.

[24] Christopher Clark, Keir Fraser, Steven Hand, Jacob Gorm Hansen, Eric Jul, Christian Limpach, Ian Pratt, and Andrew Warfield. Live migration of virtual machines. In *Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation - Volume 2*, NSDI'05, pages 273–286, Berkeley, CA, USA, 2005. USENIX Association.

[25] University of Campinas Computer Networks Laboratory, Institute of Computing. Sinergycloud: Simulation for energy in cloud computing, 2017.

[26] Intel Corporation. Enhanced intel speedstep technology for the intel pentium m processor. Technical Whitepaper, 2004.

[27] Robert J. Creasy. The origin of the vm/370 time-sharing system. *IBM Journal of Research and Development*, 25(5):483–490, 1981.

[28] Rodrigo A. C. da Silva and Nelson L. S. da Fonseca. Topology-aware virtual machine placement in data centers. *Journal of Grid Computing*, 14(1):75–90, Mar 2016.

[29] Gargi Dasgupta, Amit Sharma, Akshat Verma, Anindya Neogi, and Ravi Kothari. Workload management for power efficiency in virtualized data centers. *Commun. ACM*, 54:131–141, July 2011.

[30] H. H. de Paula Moraes Costa, A. P. F. de Araújo, J. J. C. Gondim, M. T. de Holanda, and M. E. M. T. Walter. Attribute based access control in federated clouds: A case study in bionformatics. In *2017 12th Iberian Conference on Information Systems and Technologies (CISTI)*, pages 1–7, June 2017.

[31] Lien Deboosere, Bert Vankeirsbilck, Pieter Simoens, Filip De Turck, Bart Dhoedt, and Piet Demeester. Efficient resource management for virtual desktop cloud computing. *The Journal of Supercomputing, Feb, 14, 2012*, pages 1–27, 2012.

[32] Michael Dell. Setting a higher bar for climate change. Forbes Magazine. https://www.forbes.com/2009/12/02/copenhagen-energy-efficiency-technology-cio-network-michael-dell.html#77b4498bec01, 2009. Acessado em 15/01/2018.

[33] Larry Dignan. Aws cloud computing ops, data centers, 1.3 million servers creating efficiency flywheel. ZDNet. http://www.zdnet.com/article/porsche-considers-going-all-electric-in-the-future/, 2016.

[34] Official Ubuntu Documentation. Ubuntu server (cli) installation recommended minimum system requirements. `https://help.ubuntu.com/community/Installation/SystemRequirements`, 2014.

[35] Felipe Aparecido dos Santos Novais, Lúcio Agostinho Rocha, and Fábio Luciano Verdi. Vm2t — um sistema para auxílio na migração de vms em nuvens openstack. *XXXIII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, may 2015.

[36] Truong Vinh Truong Duy, Y. Sato, and Y. Inoguchi. Performance evaluation of a green scheduling algorithm for energy savings in cloud computing. In *IEEE International Symposium on Parallel Distributed Processing, Workshops and Phd Forum (IPDPSW), 2010*, pages 1 –8, abril 2010.

[37] Patricia Takako Endo, Glauco Estácio Gonçalves, Daniel Rosendo, Demis Gomes, Guto Leoni Santos, André Luis Cavalcanti Moreira, Judith Kelner, Djamel Sadok, and Mozhgan Mahloo. Highly available clouds: System modeling, evaluations, and open challenges. In *Research Advances in Cloud Computing*, pages 21–53. Springer, 2017.

[38] Xiaobo Fan, Wolf-Dietrich Weber, and Luiz Andre Barroso. Power provisioning for a warehouse-sized computer. *SIGARCH Comput. Archit. News*, 35(2):13–23, June 2007.

[39] Hasanul Ferdaus, Manzur Murshed, Rodrigo N. Calheiros, and Rajkumar Buyya. *Emerging Research in Cloud Distributed Computing Systems, Chapter: Network-aware Virtual Machine Placement and Migration in Cloud Data Centers*. IGI Global, 2015.

[40] F. Fittkau, S. Frey, and W. Hasselbring. Cdosim: Simulating cloud deployment options for software migration support. In *2012 IEEE 6th International Workshop on the Maintenance and Evolution of Service-Oriented and Cloud-Based Systems (MESOCA)*, pages 37–46, Sept 2012.

[41] Soren Frey, Wilhelm Hasselbring, and Benjamin Schnoor. Automatic conformance checking for migrating software systems to cloud infrastructures and platforms. *Journal of Software: Evolution and Process*, 25(10):1089–1115, 2013.

[42] Erich Gamma and Kent Beck. Junit, 2006.

[43] Saurabh Kumar Garg, Chee Shin Yeo, Arun Anandasivam, and Rajkumar Buyya. Environment-conscious scheduling of hpc applications on distributed cloud-oriented data centers. *J. Parallel Distrib. Comput.*, 71:732–749, June 2011.

[44] Google. App engine. https://cloud.google.com/appengine/, Acessado em 15/01/2018.

[45] Greenpeace. Make it green – cloud computing and its contribution to climate change. Greenpeace International. https://goo.gl/FF8KXc, 2010.

[46] The Green Grid. Acessado em 15/01/2018. https://www.thegreengrid.org/, 2010.

[47] Stephan GroÃŸ and Alexander Schill. Towards user centric data governance and control in the cloud. In Jan Camenisch and Dogan Kesdogan, editors, *Open Problems in Network Security*, volume 7039 of *Lecture Notes in Computer Science*, pages 132–144. Springer Berlin / Heidelberg, 2012. 10.1007/978-3-642-27585-2_11.

[48] Brandon Heller, Srinivasan Seetharaman, Priya Mahadevan, Yiannis Yiakoumis, Puneet Sharma, Sujata Banerjee, and Nick McKeown. Elastictree: Saving energy in data center networks. In *Nsdi*, volume 10, pages 249–264, 2010.

[49] Michael R. Hines and Kartik Gopalan. Post-copy based live virtual machine migration using adaptive pre-paging and dynamic self-ballooning. In *Proceedings of the 2009 ACM SIGPLAN/SIGOPS international conference on Virtual execution environments*, VEE '09, pages 51–60, New York, NY, USA, 2009. ACM.

[50] Liting Hu, Hai Jin, Xiaofei Liao, Xianjie Xiong, and Haikun Liu. Magnet: A novel scheduling policy for power reduction in cluster with virtual machines. In *IEEE International Conference on Cluster Computing, 2008*, pages 13 –22, 29 2008-out. 1 2008.

[51] Po-Kuan Huang and Soheil Ghiasi. Power-aware compilation for embedded processors with dynamic voltage scaling and adaptive body biasing capabilities. In *Proc. of the Conf. on Design, Automation and Test in Europe*, DATE '06, pages 943–944, 3001 Leuven, Belgium, Belgium, 2006. European Design and Automation Association.

[52] IEEE 802.3 Ethernet Working Group. Ieee standards for local area networks: Supplements to carrier sense multiple access with collision detection (csma/cd) access method and physical layer specifications. *ANSI/IEEE Std 802.3a,b,c, and e-1988*, pages 0_1–, 1987.

[53] IEEE 802.3 Ethernet Working Group. IEEE 802.3 Industry Connections Ethernet Bandwidth Assessment. Technical report, IEEE, 2012.

[54] IEEE 802.3 Ethernet Working Group. 100 Gb/s Backplane and Copper Cable Task Force - Meeting Material - Mar 19-21, 2013 - Orlando, FL, USA. Technical report, IEEE, 2013.

[55] Alexandru Iosup and Dick Epema. Grid computing workloads. *IEEE Internet Computing*, 15(2):19–26, 2011.

[56] Alexandru Iosup, Ozan Sonmez, Shanny Anoep, and Dick Epema. The performance of bags-of-tasks in large-scale distributed systems. In *Proceedings of the 17th International Symposium on High Performance Distributed Computing*, HPDC '08, pages 97–108, New York, NY, USA, 2008. ACM.

[57] Rosy Jan, Wasim Rashid Wani, and Obaid Hafiz. Scientometric analysis of cloud computing. *DigitalCommonsUniversity of Nebraska-Lincoln Library Philosophy and Practice e-journal*, N/A(1273), 2015.

[58] Yaser Jararweh, Moath Jarrah, Zakarea Alshara, Mohammed Noraden Alsaleh, Mahmoud Al-Ayyoub, et al. Cloudexp: a comprehensive cloud computing experimental framework. *Simulation Modelling Practice and Theory*, 49:180–192, 2014.

[59] Peter Johnson and Tony Marker. Data centre energy efficiency. Technical report, Equipment Energy Efficiency Committee (E3), 2009.

[60] Mohammed Khalid Kaleem, Rituraj Jain, and Manaullah Abid Husain. Role of cloud computing in creating a sustainable green ict infrastructure. *International Journal of Computer Applications*, 160(1), 2017.

[61] Wonyoung Kim, M. S. Gupta, G. Y. Wei, and D. Brooks. System level analysis of fast, per-core dvfs using on-chip switching regulators. In *2008 IEEE 14th International Symposium on High Performance Computer Architecture*, pages 123–134, Feb 2008.

[62] D. Kliazovich, S. T. Arzo, F. Granelli, P. Bouvry, and S. U. Khan. Accounting for load variation in energy-efficient data centers. In *2013 IEEE International Conference on Communications (ICC)*, pages 2561–2566, June 2013.

[63] D. Kliazovich, S. T. Arzo, F. Granelli, P. Bouvry, and S. U. Khan. e-stab: Energy-efficient scheduling for cloud computing applications with traffic load balancing. In *2013 IEEE International Conference on Green Computing and Communications and IEEE Internet of Things and IEEE Cyber, Physical and Social Computing*, pages 7–13, Aug 2013.

[64] Dzmitry Kliazovich, Pascal Bouvry, and Samee Ullah Khan. Greencloud: a packet-level simulator of energy-aware cloud computing data centers. *The Journal of Supercomputing*, 62(3):1263–1283, 2012.

[65] Dzmitry Kliazovich, Pascal Bouvry, and Samee Ullah Khan. Dens: data center energy-efficient network-aware scheduling. *Cluster Computing*, 16(1):65–75, Mar 2013.

[66] Eric Knorr and Galen Gruman. What cloud computing really means. InfoWorld. https://www.infoworld.com/article/2683784/cloud-computing/what-is-cloud-computing.html, 2017. Acessado em 15/01/2018.

[67] Andreas Kohne, Marc Spohr, Lars Nagel, and Olaf Spinczyk. Federatedcloudsim: a sla-aware federated cloud simulation framework. In *Proceedings of the 2nd International Workshop on CrossCloud Systems*, page 3. ACM, 2014.

[68] Jonathan G. Koomey. Estimating total power consumption by servers in the U.S. and the world. Technical report, Lawrence Derkley National Laboratory, Feb 2007.

[69] Mohan Raj Velayudhan Kumar and Shriram Raghunathan. Heterogeneity and thermal aware adaptive heuristics for energy efficient consolidation of virtual machines in infrastructure clouds. *Journal of Computer and System Sciences*, pages –, 2015.

[70] Prakash Kumar, Krishna Gopal, and J Gupta. Scheduling algorithms for cloud: A survey and analysis. *Journal of Information Sciences and Computing Technologies*, 3(1):162–169, 2015.

[71] Dara Kusic, Jeffrey O. Kephart, James E. Hanson, Nagarajan Kandasamy, and Guofei Jiang. Power and performance management of virtualized computing environments via lookahead control. In *Proceedings of the 2008 International Conference on Autonomic Computing*, ICAC '08, pages 3–12, Washington, DC, USA, 2008. IEEE Computer Society.

[72] Daniel Lago, Edmundo Madeira, and Luiz Bittencourt. Power-aware virtual machine scheduling on clouds using active cooling control and DVFS. In *Proceedings of the 9th International Workshop on Middleware for Grids, Clouds and e-Science*, MGC '11, pages 2:1–2:6, 2011.

[73] Daniel Lago, Edmundo Madeira, and Deep Medhi. High speed network impacts and power consumption estimation for cloud data centers. In *Proceedings of the 30th Annual ACM Symposium on Applied Computing*, SAC '15, pages 615–620, New York, NY, USA, 2015. ACM.

[74] Daniel Lago, Edmundo Madeira, and Deep Medhi. On makespan, migrations, and qos workloads' execution times in high speed data centers. *IEICE Transactions*, 98-B(11):2099–2110, 2015.

[75] Daniel Lago, Edmundo Madeira, and Deep Medhi. Energy-aware virtual machine scheduling on data centers with heterogeneous bandwidths. *IEEE Transactions on Parallel and Distributed Systems*, 29(1):83–98, Jan 2018.

[76] Daniel Lago, Edmundo RM Madeira, and Luiz Fernando Bittencourt. Escalonamento com prioridade na alocaçao ciente de energia de máquinas virtuais em nuvens (in portuguese). *XXX Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, pages 508–521, abr 2012.

[77] Junghoon Lee, Gyung-Leen Park, and Hye-Jin Kim. Multithreaded power consumption scheduler based on a genetic algorithm. In Tai-hoon Kim, Hojjat Adeli, Wai-chi Fang, Thanos Vasilakos, Adrian Stoica, Charalampos Z. Patrikakis, Gansen Zhao, Javier García Villalba, and Yang Xiao, editors, *Communication and Networking*, volume 265 of *Communications in Computer and Information Science*, pages 47–52. Springer Berlin Heidelberg, 2012.

[78] Laurent Lefèvre and Jean-Marc Pierson. Energy savings in ict and ict for energy savings. *ERCIM NEWS number 79, Towards Green ICT, pg. 10411*, 2009.

[79] Haikun Liu, Cheng-Zhong Xu, Hai Jin, Jiayu Gong, and Xiaofei Liao. Performance and energy modeling for live migration of virtual machines. In *Proceedings of the 20th international symposium on High performance distributed computing*, HPDC '11, pages 171–182, New York, NY, USA, 2011. ACM.

[80] Debendra Maharana, Bibhudatta Sahoo, and Srinivas Sethi. Energy-efficient real-time tasks scheduling in cloud data centers. *IJSEAT*, 4(12):768–773, 2017.

[81] Makoto Sakai (Tokyo, JP). Acpi sleep control, July 2001. Patent Number 6266776.

[82] Rahul Malhotra and Prince Jain. Study and comparison of various cloud simulators available in the cloud computing. *International Journal*, 3(9), 2013.

[83] Andrew McAfee. 2010: The year the cloud rolled in. Harvard Business Review. https://hbr.org/2010/12/2010-the-year-the-cloud-rolled, 2010.

[84] Peter Mell and Tim Grance. The nist definition of cloud computing. *National Institute of Standards and Technology*, 53(6):50, 2009.

[85] Xiaoqiao Meng, Vasileios Pappas, and Li Zhang. Improving the scalability of data center networks with traffic-aware virtual machine placement. In *INFOCOM, 2010 Proceedings IEEE*, pages 1–9, 2010.

[86] Microsoft. Windows azure plataform. https://azure.microsoft.com/en-us/?v=18.01, Acessado em 15/01/2018.

[87] Anand Motwani, Vaibhav Patel, and Vijay Patil. Power and qos aware virtual machine consolidation in green cloud data center. *International Journal of Electrical, Electronics and Computer Engineering*, 1(4):93–96, 2015.

[88] S. Murugesan. Harnessing green it: Principles and practices. *IT Professional*, 10(1):24 –33, jan.-fev. 2008.

[89] Ripal Nathuji and Karsten Schwan. Virtualpower: coordinated power management in virtualized enterprise systems. *SIGOPS Oper. Syst. Rev.*, 41:265–278, October 2007.

[90] Sergiu Nedevschi, Lucian Popa, Gianluca Iannaccone, Sylvia Ratnasamy, and David Wetherall. Reducing network energy consumption via sleeping and rate-adaptation. In *Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation*, NSDI'08, pages 323–336, Berkeley, CA, USA, 2008. USENIX Association.

[91] Alberto Núñez, Jose L Vázquez-Poletti, Agustin C Caminero, Gabriel G Castañé, Jesus Carretero, and Ignacio M Llorente. icancloud: A flexible and scalable cloud infrastructure simulator. *Journal of Grid Computing*, 10(1):185–209, 2012.

[92] Institute of Electrical and Electronics Engineers. The 2012 ieee fifth international conference on cloud computing (cloud 2012), 2012.

[93] OECD Working Party on the Information Economy. Towards green ict strategies: Assessing policies and programmes on icts and the environment. Organisation for Economic Co-operation and Development. http://www.oecd.org/internet/ieconomy/42825130.pdf, 2009. Acessado em 15/01/2018.

[94] OpenNebula. Data center federation overview. https://goo.gl/bqbRZc, 2018. Acessado em 15/01/2018.

[95] OpenStack. Scheduling. https://docs.openstack.org/mitaka/config-reference/compute/scheduler.html, Acessado em 15/01/2018.

[96] Simon Ostermann, Kassian Plankensteiner, Radu Prodan, and Thomas Fahringer. Groudsim: an event-based simulation framework for computational grids and clouds. In *European Conference on Parallel Processing*, pages 305–313. Springer, 2010.

[97] Michael Pawlish, Aparna S Varde, and Stefan A Robila. Analyzing utilization rates in data centers for optimizing energy management. In *Green Computing Conference (IGCC), 2012 International*, pages 1–6. IEEE, 2012.

[98] Jing Tai Piao and Jun Yan. A network-aware virtual machine placement and migration approach in cloud computing. In *Grid and Cooperative Computing (GCC), 2010 9th International Conference on*, pages 87–92, 2010.

[99] Jian ping Luo, Xia Li, and Min rong Chen. Hybrid shuffled frog leaping algorithm for energy-efficient dynamic consolidation of virtual machines in cloud data centers. *Expert Systems with Applications*, 41(13):5804 – 5816, 2014.

[100] H. Qian, F. Li, R. Ravindran, and D. Medhi. Optimal resource provisioning and the impact of energy-aware load aggregation for dynamic temporal workloads in data centers. *IEEE Transactions on Network and Service Management*, 11(4):486–503, Dec 2014.

[101] Haiyang Qian, Fu Li, and D. Medhi. On energy-aware aggregation of dynamic temporal demand in cloud computing. In *Communication Systems and Networks (COMSNETS), 2012 Fourth International Conference on*, pages 1 –6, jan. 2012.

[102] Jan M. Rabaey. *Digital integrated circuits: a design perspective*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1996.

[103] Aboozar Rajabi, Hamid Reza Faragardi, and Thomas Nolte. An efficient scheduling of hpc applications on geographically distributed cloud data centers. In *International Symposium on Computer Networks and Distributed Systems*, pages 155–167. Springer, 2013.

[104] Christian Esteve Rothenberg. Re-architected cloud data center networks and their impact on the future internet. In *New Network Architectures*, pages 179–187. Springer, 2010.

[105] G. Semeraro, G. Magklis, R. Balasubramonian, and D.H. Albonesi et al. Energy-eficient processor design using multiple clocks domains with dynamic voltage and frequency scaling. In *Proc. 8th Intl. Symp. on High-Performance Computer Architecture*, pages 29–40, Feb 2002.

[106] Arman Shehabi, Sarah Josephine Smith, Dale A. Sartor, Richard E. Brown, Magnus Herrlin, Jonathan G. Koomey, Eric R. Masanet, Nathaniel Horner, Inês Lima Azevedo, and William Lintner. United states data center energy usage report. *Lawrence Berkeley National Laboratory, Berkeley, California. LBNL-1005775*, 06/2016 2016.

[107] Network Simulator. The network simulator–ns-2, 2009.

[108] Ankit Singla, P. Brighten Godfrey, and Alexandra Kolla. High throughput data center topology design. In *Proceedings of the 11th USENIX Conference on Networked Systems Design and Implementation*, NSDI'14, pages 29–41, Berkeley, CA, USA, 2014. USENIX Association.

[109] Shekhar Srikantaiah, Aman Kansal, and Feng Zhao. Energy aware consolidation for cloud computing. In *Proceedings of the 2008 conference on Power aware computing and systems*, HotPower'08, pages 10–10, Berkeley, CA, USA, 2008. USENIX Association.

[110] Ilango Sriram. *SPECI, a Simulation Tool Exploring Cloud-Scale Data Centres*, chapter Full Papers, pages 381–392. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[111] Alexander Stage and Thomas Setzer. Network-aware migration control and scheduling of differentiated virtual machine workloads. In *Proceedings of the 2009 ICSE Workshop on Software Engineering Challenges of Cloud Computing*, CLOUD '09, pages 9–14, Washington, DC, USA, 2009. IEEE Computer Society.

[112] Standards I. E. E. E. Association. ANSI IEEE 802.3 Standard. Technical report, IEEE, May 1998.

[113] Yevgeniy Sverdlik. Hereś how much energy all us data centers consume. Data Center Knowledge. https://goo.gl/4aFdyg, 2016.

[114] Cisco Systems. Unified fabric: Cisco's innovation for data center networks. Technical Whitepaper, 2009.

[115] Cheng-Jen Tang, Miau-Ru Dai, Hui-Chin He, and Chi-Cheng Chuang. Evaluating energy efficiency of data centers with generating cost and service demand. *Bulletin of Networking, Computing, Systems, and Software*, 1(1), 2012.

[116] Thiago Teixeira Sá, Rodrigo N. Calheiros, and Danielo G. Gomes. *CloudReports: An Extensible Simulation Tool for Energy-Aware Cloud Computing Environments*, chapter 6, pages 127–142. Springer International Publishing, Cham, 2014.

[117] M. Tighe, G. Keller, M. Bauer, and H. Lutfiyya. Dcsim: A data centre simulation tool for evaluating dynamic virtualized resource management. In *2012 8th international conference on network and service management (cnsm) and 2012 workshop on systems virtualiztion management (svm)*, pages 385–392, Oct 2012.

[118] TIOBE. Tiobe index. https://www.tiobe.com/tiobe-index/, 2017.

[119] Luis M. Vaquero, Luis Rodero-Merino, Juan Caceres, and Maik Lindner. A break in the clouds: towards a cloud definition. *SIGCOMM Comput. Commun. Rev.*, 39(1):50–55, December 2008.

[120] Akshat Verma, Puneet Ahuja, and Anindya Neogi. pmapper: Power and migration cost aware application placement in virtualized systems. In Valerie Issarny and Richard Schantz, editors, *Middleware 2008*, volume 5346 of *Lecture Notes in Computer Science*, pages 243–264. Springer Berlin Heidelberg, 2008.

[121] David Villegas, Norman Bobroff, Ivan Rodero, Javier Delgado, Yanbin Liu, Aditya Devarakonda, Liana Fong, S. Masoud Sadjadi, and Manish Parashar. Cloud federation in a layered service model. *Journal of Computer and System Sciences*, January 2012.

[122] VMware. vmotion for live migration of virtual machines without service interruption. https://www.vmware.com/pdf/vmotion_datasheet.pdf, 2010. Acessado em 15/01/2018.

[123] VMware. Vmware distributed power management: Concepts and usage. https://www.vmware.com/techpapers/2008/vmware-distributed-power-management-concepts-and-1080.html, 2013. Acessado em 15/01/2018.

[124] William Voorsluys, James Broberg, Srikumar Venugopal, and Rajkumar Buyya. Cost of virtual machine live migration in clouds: A performance evaluation. In *Proceedings of the 1st International Conference on Cloud Computing*, CloudCom '09, pages 254–265, Berlin, Heidelberg, 2009. Springer-Verlag.

[125] D. Wang. Meeting green computing challenges. In *International Symposium on High Density packaging and Microsystem Integration, 2007. HDP '07*, pages 1–4, june 2007.

[126] Reinhold P Weicker. Dhrystone benchmark: rationale for version 2 and measurement rules. *ACM SIGPLAn notices*, 23(8):49–62, 1988.

[127] Bhathiya Wickremasinghe, Rodrigo N Calheiros, and Rajkumar Buyya. Cloudanalyst: A cloudsim-based visual modeller for analysing cloud computing environments and applications. In *Advanced Information Networking and Applications (AINA), 2010 24th IEEE International Conference on*, pages 446–452. IEEE, 2010.

[128] Timothy Wood, Prashant Shenoy, Arun Venkataramani, and Mazin Yousif. Black-box and gray-box strategies for virtual machine migration. In *Proceedings of the 4th*

USENIX conference on Networked systems design & implementation, NSDI'07, pages 17–17, Berkeley, CA, USA, 2007. USENIX Association.

[129] Meng Xu, Lizhen Cui, Haiyang Wang, and Yanbing Bi. A multiple qos constrained scheduling strategy of multiple workflows for cloud computing. In *IEEE International Symposium on Parallel and Distributed Processing with Applications, 2009*, pages 629 –634, ago. 2009.

[130] Weilian Xue, Wenxin Li, Heng Qi, Keqiu Li, Xiaoyi Tao, and Xinhua Ji. Communication-aware virtual machine migration in cloud data centres. *International Journal of High Performance Computing and Networking*, 10(4-5):372–380, 2017.

[131] Rerngvit Yanggratoke, Fetahi Wuhib, and Rolf Stadler. Gossip-based resource allocation for green computing in large clouds. In *7th International Conference on Network and Service Management (CNSM), 2011*, pages 1 –9, out. 2011.

[132] Qi Zhang, Lu Cheng, and Raouf Boutaba. Cloud computing: state-of-the-art and research challenges. *Journal of Internet Services and Applications*, 1(1):7–18, 2010.

[133] Zhiming Zhang, Chan-Ching Hsu, and Morris Chang. Cool cloud: A practical dynamic virtual machine placement framework for energy aware data centers. In *Cloud Computing (CLOUD), 2015 IEEE 8th International Conference on*, pages 758–765, June 2015.

[134] Wei Zhao, Yong Peng, Feng Xie, and Zhonghua Dai. Modeling and simulation of cloud computing: A review. In *Cloud Computing Congress (APCloudCC), 2012 IEEE Asia Pacific*, pages 20–24. IEEE, 2012.

[135] Ao Zhou, Shangguang Wang, Chenchen Yang, Lei Sun, Qibo Sun, and Fangchun Yang. Ftcloudsim: support for cloud service reliability enhancement simulation. *International Journal of Web and Grid Services*, 11(4):347–361, 2015.

Apêndice A

SinergyCloud

Para poder avaliar o `TEA`, inicialmente cogitamos, por razões históricas, usar o simulador *CloudSim*. No entanto, nos deparamos com uma série de limitações deste simulador, incluindo, mas não se limitando a: questões de escalabilidade para cenários complexos — o simulador apresenta muitos recursos que demandam processamento fora do nosso escopo de análise; impossibilidade de realizar estudos de consumo de energia dos elementos de núcleo da rede; impossibilidade de usar modelos de potência em todos os elementos de núcleo e de borda; dificuldades em mensurar abortamentos de migrações de máquinas virtuais em virtude da finalização destas; uso padrão de topologias com armazenamento de máquinas virtuais em servidores e não em *storages*. Em virtude disso, fizemos uma extensa pesquisa sobre simuladores (conforme Seção 3), mas não encontramos nenhum simulador que fosse capaz de possibilitar satisfatoriamente as análises desejáveis para mensurar o desempenho do `TEA`. Também consideramos modificar o CloudSim para suportar todos os recursos desejados, mas após minuciosa análise técnica consideramos ser tecnicamente mais viável a criação de um novo simulador que possibilitasse tais análises, não só para este trabalho, mas para uso geral da comunidade científica.

Diante do exposto, optamos por desenvolver o *SinergyCloud* [25], que é um *toolkit* de simulação que foca na avaliação do comportamento de *data centers* nos processos de escalonamento de máquinas virtuais e de cargas de trabalho. Esta ferramenta, desenvolvida em Java, orientada a eventos e em nível de pacotes tem como objetivo analisar consumo de energia, *makespan*, tempo de finalização e de processamento de cargas de trabalho, entre outras características desejáveis para este trabalho.

A.1 Introdução

Entre as muitas abordagens voltadas para a economia de energia em *data centers* está o escalonamento ciente de energia de máquinas virtuais e de cargas de trabalho [80]. Alguns fatores dos *data centers*, tais como tamanho; incapacidade de desligar máquinas em funcionamento; a necessidade de grandes modificações na estrutura do *data center* para testes e pesquisas; entre outros; tornam irrealizável a implantação direta de sistemas de escalonamentos inovadores. Para ligar com estas limitações, relacionadas com a rigidez

da infraestrutura, para avaliação do desempenho das aplicações sob condições variáveis, o uso de simuladores é comumente empregado como ferramenta crucial para pesquisa.

Aqui apresentamos o *SinergyCloud*, um *framework* para modelar e simular o ambiente da computação em nuvem, com foco principal no escalonamento, consumo de energia, *makespan* e tempo de execução de cargas, com o objetivo de diminuir a complexidade nos testes de sistemas de computação em nuvem. O objetivo do *SinergyCloud* é permitir desenvolvedores e pesquisadores analisarem o comportamento de nuvens computacionais, com granularidade dinâmica, de modo fácil e com uma menor curva de aprendizado que outros simuladores, sem se preocuparem com detalhes relacionados com a infraestrutura das nuvens, abrindo a possibilidade de avaliar hipóteses em um ambiente controlado.

É possível editar com pouco esforço eventos, por exemplo, como uma conexão é estabelecida; como uma migração de máquina virtual é iniciada e terminada, qual é o servidor de origem, servidor de destino e o *broker* responsável; como uma carga de trabalho começa a ser submetida pelo *broker*, processada e retornada pela máquina virtual; o que fazer quando um pacote de uma dada conexão atinge um determinado *switch*; etc. Também é possível operar em vários níveis da arquitetura, como criar modelos de consumo de potência, algoritmos de escalonamento de máquinas virtuais, algoritmos de escalonamento de cargas de trabalho, algoritmos de armazenamento de imagens de máquinas virtuas, algoritmos personalizados de STP, etc.

O *SinergyCloud*, portanto, surge como uma opção ao número relativamente limitado de simuladores para a computação em nuvem [82], em especial àqueles que são capazes de lidar com escalonamento e consumo de energia, apresentando uma arquitetura flexível e facilmente estendível para adaptar às necessidades do pesquisador. Nós observamos que, para o melhor de nosso conhecimento, não conseguimos encontrar um simulador capaz de trabalhar no contexto de IaaS com o objetivo de permitir analisar o processo de escalonamento na computação em nuvem, desenvolvido em uma linguagem de programação amplamente utilizada, focado na análise do consumo de energia da nuvem, sendo esta nuvem constituída de *data centers* individuais ou geodistribuídos, e incluindo nesta análise o consumo de energia tanto de elementos de borda quanto elementos de núcleo.

A.2 Características

O *SinergyCloud* é um *framework* que possibilita a execução de simulações suaves e contínuas, para um amplo espectro de cenários, focado principalmente no consumo de energia e na computação em nuvem em nível da infraestrutura. Este *toolkit* de simulação foi desenvolvido para lidar com o problema de avaliação de desempenho em ambientes distribuídos, possuindo como características principais:

- Desenvolvido em Java → este *framework* de simulação é construído sobre a linguagem de programação Java, a primeira colocada no ranking de uso TIOBE [118];
- Dirigido a Eventos → Simulações de Eventos Discretos (DES) são um tipo de simulação onde eventos são mantidos em uma fila ordenada pelo tempo e processados em um dado tempo de simulação [134]. O *SinergyCloud* apresenta todos os recursos de

um simulador orientado a eventos: ele possui variáveis internas que mantêm registro do tempo de simulação; o estado do sistema (/seus componentes) é armazenado; existem pontos que indicam a mudança de estado do sistema, e cada evento possui seu tempo associado. O motor trabalha sobre um conjunto de eventos ordenado por tempo, de modo que eles são processados enquanto a simulação não estiver concluída;

- Nível de Pacotes → Diferentemente de simuladores de computação em nuvem que trabalham somente com grafos direcionados que indicam o fluxo de dados nas redes, o *SinergyCloud* implementa tráfego em nível de pacote. Cada pacote em trânsito é representado por um objeto com estrutura abstraída do formato de quadros da arquitetura TCP/IP. Nativamente, o *SinergyCloud* provê suporte para os seguintes formatos de quadros: *Ethernet* no nível de enlace; IPv4 e IPv6 no nível de rede; e TCP e UDP no nível de transporte.

Este *framework* de simulação provê classes básicas para descrever um ambiente de nuvem, incluindo *data centers*, máquinas virtuais, recursos computacionais, políticas de escalonamento e provisionamento de diversas partes do sistema, etc. Estes componentes podem ser usados para avaliar novas topologias, algoritmos de escalonamento de máquinas virtuais e cargas de trabalho, etc., com foco principal no consumo de energia, *makespan* e tempo de processamento de cargas de trabalho individuais. Este *toolkit* possibilita construir cenários simplesmente por estender ou substituir as classes existentes e codificando o cenário desejado.

A.3 Arquitetura

As principais funções do *SinergyCloud* são implementadas em um pacote Core. Nele existem abstrações configuráveis pelo usuário para executar simulações, tais como *data centers* — constituídos de *brokers*, *storages*, servidores, *switches*, roteadores de borda, SLAs, máquinas virtuais e cargas de trabalho — bem como a implementação de componentes de núcleo, como geodistribuição de *data centers*, conexões, pacotes de rede e eventos.

Apresentamos na Figura A.1 um relacionamento entre estes componentes, que são:

- Carga de Trabalho → representa uma carga a ser processada;
- Máquina Virtual → processa cargas de trabalho;
- SLA → restrição de uma máquina virtual que deve ser atendida pela nuvem;
- Servidor → servidor responsável pelo provisionamento de recursos (ex: processamento, RAM, etc.) para às máquinas virtuais;
- *Storage* → responsável por armazenar imagens das máquinas virtuais;

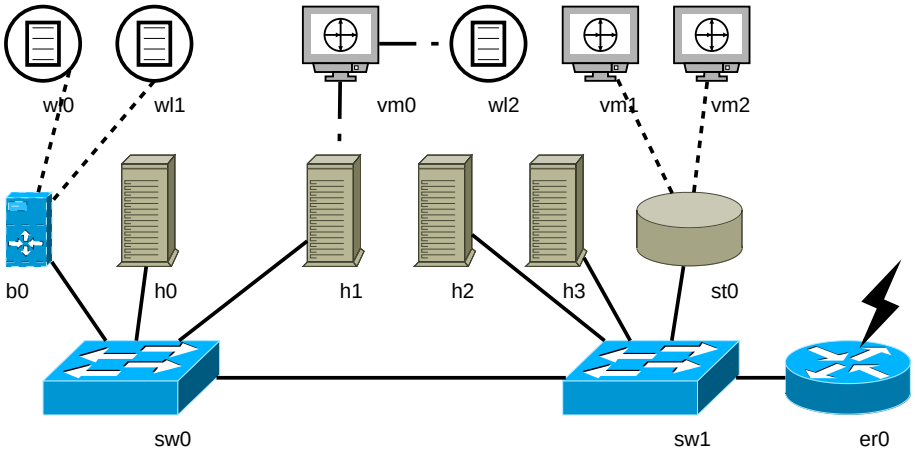

Figura A.1: *SinergyCloud*: Relacionamento entre Componentes

- *Broker* → responsável por selecionar servidores para máquinas virtuais, enviar cargas de trabalho para máquinas virtuais operacionais, e por receber cargas processadas;
- *Switch* → interconecta nós físicos de rede, que podem ser roteadores de borda, *switches*, *brokers*, *storages* ou servidores;
- Roteador de Borda → permite a interconexão de *data centers*.

Um cenário de simulação usualmente é constituído de pelo menos um *broker*, um *storage*, um servidor, de modo que todos estes elementos estão interconectados. Nós apresentamos na Figura A.2 um esquema destes elementos de núcleo e suas interações.

O *SinergyCloud* usa uma simulação orientada a eventos. Esta decisão de implementação traz muitas vantagens de desempenho comparada aos simuladores orientados a passos de tempo, pois o *SinergyCloud* pode concentrar um número grande de operações a serem realizadas e uma única vez, o que não seria possível se a segunda implementação fosse utilizada.

Para demonstrar esta vantagem da orientação a eventos, veja o seguinte exemplo: suponha que um servidor, que possui somente uma máquina virtual, recebe uma carga de trabalho grande (que levaria, por exemplo, 5 horas para concluir). Um simulador orientado a passos de tempo, com — por exemplo — intervalos de processamento de 1 segundo, precisaria, neste caso, de realizar atualizações dos cálculos de processamento desta carga de trabalho $5 \times 60 \times 60 \times 1 = 18.000$ vezes. No entanto, no caso da orientação a eventos, como nada ocorre neste intervalo entre o recebimento da carga pela máquina virtual e a conclusão deste processamento, somente um evento é escalonado, para o tempo estimado de conclusão do processamento desta carga, portanto, somente 1 processamento seria necessário (contra 18.000 requeridos pela simulação orientada a passos de tempo).

Nós enfatizamos, porém, que se o usuário desejar uma simulação orientada a passos de tempo, é possível, gerando eventos com as atualizações desejadas, para os intervalos que o usuário desejar.

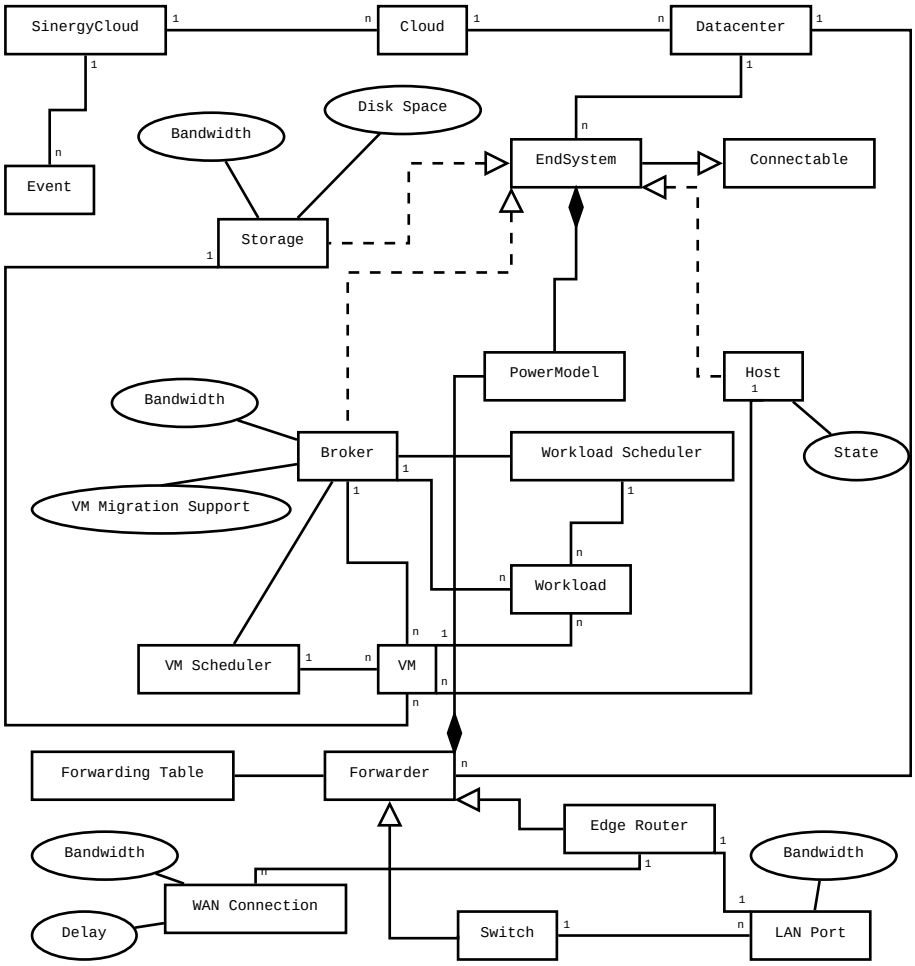

Figura A.2: *SinergyCloud*: Arquitetura do Núcleo

Nós apresentamos na Tabela A.1 os eventos suportados pelo *SinergyCloud*. É possível ao usuário do *SinergyCloud* personalizar qualquer um deles, adicionando recursos necessários para a sua pesquisa.

O motor de simulação permite ao usuário acessar as seguintes informações: indicação de quando a simulação foi iniciada, número de desligamentos de servidores, número de ligações de servidores, número de desligamentos de *switches*, número de ligações de *switches*, número de migrações iniciadas, número de migrações concluídas, número de migrações abortadas, e uma lista contendo os tempos de execução de cada migração. O motor de simulador também executa um STP nos *data centers* em um número personalizado de tempo, por padrão, 30 segundos.

As principais funções do motor de simulação são *Set Up* e *Run*.

Set Up Configura o *SinergyCloud* para realizar uma nova simulação. Alguns parâmetros que podem ser definidos pelo usuário são: *data centers*, máquinas virtuais, cargas de trabalho, armazenador de imagens de máquinas virtuais, gerador de STP, gerador de relatórios, tamanho de *overhead* da camada de enlace e tamanho do cabeçalho das camadas de rede e de transporte. O *SinergyCloud* também permite o uso de uma heurística de

Tabela A.1: *SinergyCloud*: Eventos Suportados

Elemento	Ações	Quando
Broker	Iniciar; Atualizar	
Conexão	Estabelecer; Finalizar; Abortar; Atualizar	
Dispositivo	Alterar Estado	
Pacote	Chegar ao próximo nó; Partir de uma porta WAN	
Máquina Virtual	Iniciar	Antes/Após envio
Máquina Virtual	Migração	Iniciar; Finalizar; Abortar
Máquina Virtual	Finalizar	Antes/Após recebimento
Carga de Trabalho	Iniciar	Antes/Após envio pelo *Broker*
Carga de Trabalho	Finalizar	Antes/Após recebimento pelo *Broker*
Carga de Trabalho	Atualizar/Finalizar Processamento	

agrupamento de pacotes, que pode reduzir o tempo de execução de uma simulação tendo como *trade-off* uma pequena perda de acurácia nas durações das conexões.

Run A simulação pode ser executada após a configuração. O *SinergyCloud* basicamente verifica algumas condições iniciais para tornar a simulação possível e, uma vez atendidas, inicializa o gerador de relatórios e executa os eventos de simulação enquanto os *brokers* ainda tiverem máquinas virtuais ou cargas de trabalho para processar. Após os trabalhos dos *brokers* serem terminados, a simulação finaliza a contabilização do consumo de energia e na sequência um relatório é criado pelo gerador de relatórios especificado pelo usuário. A medida em que a simulação é executada, um *logger* imprime o progresso da simulação.

Funções secundárias usadas pelo *SinergyCloud* são implementadas fora do pacote Core. Entre estas funções podemos citar geradores de árvore geradora, modelos de consumo de potência, geradores de relatórios, armazenadores de imagens de máquinas virtuais, algoritmos de escalonamento de máquinas virtuais e utilitários.

O *SinergyCloud* acompanha algoritmos padrões de geradores de árvore geradora (com propagação em largura), armazenadores de imagens de máquinas virtuais (*round-robin*) e escalonadores de cargas de trabalho (round-robin). Nativamente, o *SinergyCloud* provê suporte aos modelos de consumo de potência Simples — considera somente duas potências: uma para o estado ligado e outra para o estado desligado — DVFS linear e DVFS stepped. Os geradores de relatório que acompanham o *SinergyCloud* permitem gerar relatórios em tela ou em arquivos CSV — com possível geração de plt para *gnuplot*. Com relação aos algoritmos de escalonamento de máquinas virtuais, o *SinergyCloud* implementa nativamente o `RND`, `RR`, `FA`, `HU`, `HR`, `MPD`, `LA` e `BALA`. Quanto aos utilitários, o *SinergyCloud* possui uma calculadora com utilidades para o escalonamento, um gerador

automatizado de elementos (*brokers*, *storages*, servidores, processadores, máquinas virtuais e cargas de trabalho), e um gerador automatizado para as topologias *Single-Hop Star*, *Flat-Tree*, *Binary Tree* e *K-Ary Fat Tree*.

O *framework SinergyCloud* foi construído em uma metodologia de desenvolvimento de especificações padrão estável usando uma abordagem de testes e construção disciplinada e rigorosa, baseado na prática Test-Driven Development (TDD). O uso desta metodologia é capaz de reduzir drasticamente as taxas de defeito do *framework*, além de manter os casos de testes unitários criados como um ativo reusável e extensível de modo a continuar provendo qualidade por toda a sua vida.

Numerosos testes foram escritos para garantir a correção e qualidade do *framework* de simulação, seguidos por suas soluções, e estes procedimentos foram realizados em sucessivas iterações. De fato, milhares de asserções foram codificadas de modo a manter sua robustez. Cada um destes testes foi cuidadosamente planejado, baseado em modelos formais, e codificado no renomado *framework* de testes JUnit [42], e o *toolkit* foi capaz de satisfazer todas a asserções programadas.

Os testes foram escritos para todos os aspectos encontrados no *framework* de simulação. Cada teste consistiu na verificação de saídas esperadas possíveis para a simulação. Em outras palavras, nós modelamos com detalhes o que era esperado para o *framework* de simulação para uma saída real e verdadeira, codificada nos testes, e passo-a-passo cada ação da simulação. No total, milhares de testes e asserções foram escritos, de modo a avaliar todos os componentes do *framework* do simulação, tanto para operações unitárias quanto integradas, e estão disponíveis para download no site do *SinergyCloud*.

Agora nós comparamos na Tabela A.2 o *SinergyCloud* a dois simuladores populares capazes de estimar o consumo de energia em nuvens computacionais: *CloudSim* [22] e *GreenCloud* [64].

Tabela A.2: Comparação Qualitativa entre *CloudSim*, *GreenCloud* e *SinergyCloud*

	CloudSim	GreenCloud	SinergyCloud
Disponibilidade	GPL		
Platforma	Java	NS2 (C++ & Otcl)	Java
Tempo de Simulação	Rápido	Lento	Médio
Suporte Gráfico	GUI Limitado		
Modelo de Comunicação	Orientado a Eventos	Pilha TCP/IP Completa	Nível de Pacotes
Modelos de Potência	Servidores	Servidores e Rede	
Modos de Economia de Energia (desligamento e DVFS)	Servidores	Servidores e Rede	

O *CloudSim* é um *toolkit* para modelagem e simulação de ambientes de computação em nuvem e avaliação de algoritmos de provisionamento de recursos e provisionamento, desenvolvido pela Universidade de Melbourne, Austrália, construído em cima do simulador

GridSim [20]. Do outro lado, o *GreenCloud* é um simulador de nível de pacotes de *data centers* cientes de energia, encabeçado pela University of Luxembourg, desenvolvido como uma extensão do NS2 [107]. O *SinergyCloud* é um *framework* de simulação para analisar o consumo de energia e tempos (*makespan* e tempo de execução de cargas de trabalho), considerando a energia despendida pelos elementos de núcleo da rede, focado na escalabilidade, construído a partir do zero para esta finalidade.

Em resumo, o *CloudSim* provê bons tempos de simulação enquanto o *GreenCloud* oferece mais acurácia. Com relação ao consumo de energia, o *CloudSim* tem como grande limitação a incapacidade de prover suporte ao consumo de energia do núcleo da rede, o que é uma grande limitação. Para o *GreenCloud*, entre suas principais limitações estão sua baixa escalabilidade e a linguagem de programação pouco utilizada (OTcl). Para prover uma solução para esta lacuna, o *SinergyCloud* provê bons tempos de simulação, tornando possível analisar o uso do núcleo da rede, é escalável, e usa uma linguagem de programação largamente utilizada (Java).

O *framework SinergyCloud*, sua documentação e recursos afins estão disponíveis para download através do endereço `www.lrc.ic.unicamp.br/sinergycloud/`.